

---

# AWS SDK Code Examples

## Code Library



## AWS SDK Code Examples: Code Library

Copyright © 2022 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

## Table of Contents

What is the code example library? .....	viii
Working with AWS SDKs .....	1
Code examples by service .....	1
ACM .....	3
Actions .....	4
Scenarios .....	5
API Gateway .....	16
Actions .....	22
Scenarios .....	23
Cross-service examples .....	41
API Gateway Management API .....	46
Actions .....	49
Application Auto Scaling .....	49
Actions .....	50
Application Recovery Controller .....	51
Actions .....	51
Audit Manager .....	52
Actions .....	55
Aurora .....	55
Actions .....	63
Scenarios .....	63
Cross-service examples .....	97
AWS Batch .....	126
Actions .....	129
AWS CloudFormation .....	129
Cross-service examples .....	130
CloudFront .....	130
Actions .....	131
CloudWatch .....	131
Actions .....	135
Scenarios .....	178
CloudWatch Events .....	185
Actions .....	185
CloudWatch Logs .....	192
Actions .....	192
Cross-service examples .....	210
Amazon Cognito Identity .....	211
Actions .....	211
Cross-service examples .....	213
Amazon Cognito Identity Provider .....	215
Actions .....	215
Scenarios .....	253
Amazon Cognito Sync .....	274
Actions .....	274
Amazon Comprehend .....	276
Actions .....	276
Scenarios .....	300
Cross-service examples .....	313
AWS Config .....	316
Actions .....	316
Device Farm .....	318
Scenarios .....	319
DynamoDB .....	325

Actions .....	326
Scenarios .....	449
Cross-service examples .....	594
Amazon EBS .....	603
Actions .....	603
Amazon EC2 .....	605
Actions .....	608
Scenarios .....	660
Cross-service examples .....	683
Amazon EC2 Auto Scaling .....	684
Actions .....	685
Scenarios .....	713
Amazon ECR .....	739
Actions .....	739
Amazon ECS .....	740
Actions .....	741
Amazon EKS .....	742
Actions .....	743
Amazon EMR .....	744
Actions .....	744
Scenarios .....	750
Cross-service examples .....	752
EventBridge .....	753
Actions .....	753
Scenarios .....	764
Cross-service examples .....	778
AWS Glue .....	779
Actions .....	780
Scenarios .....	817
IAM .....	862
Actions .....	863
Scenarios .....	1027
Cross-service examples .....	1098
AWS IoT .....	1099
Actions .....	1099
AWS KMS .....	1101
Actions .....	1102
Scenarios .....	1146
Amazon Keyspaces .....	1151
Actions .....	1152
Scenarios .....	1161
Kinesis .....	1171
Actions .....	1171
Scenarios .....	1194
Kinesis Data Analytics .....	1197
Actions .....	1198
Data generator .....	1206
Lambda .....	1215
Actions .....	1216
Scenarios .....	1246
Cross-service examples .....	1282
Amazon Lex .....	1289
Cross-service examples .....	1289
Lookout for Vision .....	1290
Actions .....	1290
Scenarios .....	1307
MediaLive .....	1312

Actions .....	1312
MediaPackage .....	1313
Actions .....	1313
Organizations .....	1315
Actions .....	1315
Amazon Personalize .....	1331
Actions .....	1331
Amazon Personalize Events .....	1363
Actions .....	1363
Amazon Personalize Runtime .....	1367
Actions .....	1367
Amazon Pinpoint .....	1373
Actions .....	1373
Amazon Pinpoint SMS and Voice API .....	1406
Actions .....	1407
Amazon Polly .....	1410
Actions .....	1411
Scenarios .....	1427
Cross-service examples .....	1428
QLDB .....	1429
Actions .....	1429
Amazon RDS .....	1430
Actions .....	1431
Scenarios .....	1462
Cross-service examples .....	1501
Amazon RDS Data Service .....	1505
Actions .....	1505
Cross-service examples .....	1506
Amazon Redshift .....	1509
Actions .....	1509
Cross-service examples .....	1517
Amazon Rekognition .....	1518
Actions .....	1519
Scenarios .....	1572
Cross-service examples .....	1606
Route 53 .....	1612
Actions .....	1612
Route 53 domain registration .....	1613
Actions .....	1614
Scenarios .....	1621
Amazon S3 .....	1632
Actions .....	1634
Scenarios .....	1757
Cross-service examples .....	1824
S3 Glacier .....	1833
Actions .....	1834
Scenarios .....	1860
Amazon SES .....	1867
Actions .....	1868
Scenarios .....	1897
Cross-service examples .....	1912
Amazon SES API v2 .....	1923
Actions .....	1923
Amazon SNS .....	1928
Actions .....	1929
Scenarios .....	2007
Cross-service examples .....	2015

Amazon SQS .....	2020
Actions .....	2021
Scenarios .....	2084
Cross-service examples .....	2087
AWS STS .....	2090
Actions .....	2090
Scenarios .....	2097
SageMaker .....	2107
Actions .....	2107
Scenarios .....	2118
Secrets Manager .....	2122
Actions .....	2123
Scenarios .....	2140
Cross-service examples .....	2141
Step Functions .....	2142
Actions .....	2142
Cross-service examples .....	2155
AWS Support .....	2157
Actions .....	2160
Scenarios .....	2178
Systems Manager .....	2200
Actions .....	2201
Amazon Textract .....	2202
Actions .....	2202
Scenarios .....	2216
Cross-service examples .....	2218
Amazon Transcribe .....	2220
Actions .....	2220
Scenarios .....	2241
Cross-service examples .....	2248
Amazon Translate .....	2250
Actions .....	2250
Scenarios .....	2261
Cross-service examples .....	2263
Code examples by SDK .....	2267
AWS SDK for .NET .....	2267
Single-service actions and scenarios .....	2267
Cross-service examples .....	2606
SDK for C++ .....	2608
Single-service actions and scenarios .....	2608
Cross-service examples .....	2714
SDK for Go V2 .....	2715
Single-service actions and scenarios .....	2715
SDK for JavaScript V2 .....	2801
Single-service actions and scenarios .....	2801
Cross-service examples .....	2909
SDK for JavaScript V3 .....	2909
Single-service actions and scenarios .....	2910
Cross-service examples .....	3150
SDK for Java 2.x .....	3156
Single-service actions and scenarios .....	3156
Cross-service examples .....	3494
SDK for Kotlin .....	3499
Single-service actions and scenarios .....	3499
Cross-service examples .....	3663
SDK for PHP .....	3665
Single-service actions and scenarios .....	3665

Cross-service examples .....	3735
SDK for Python (Boto3) .....	3736
Single-service actions and scenarios .....	3736
Cross-service examples .....	4304
SDK for Ruby .....	4310
Single-service actions and scenarios .....	4311
SDK for Rust .....	4391
Single-service actions and scenarios .....	4391
Cross-service examples .....	4481
SDK for SAP ABAP .....	4482
Single-service actions and scenarios .....	4483
SDK for Swift .....	4557
Single-service actions and scenarios .....	4557

There are more AWS SDK examples available in the [AWS Doc SDK Examples GitHub repo](#).

# What is the code example library?

The code example library is a collection of code examples that show you how to use AWS software development kits (SDKs) with AWS.

The examples are organized by AWS service and by AWS SDK. The same examples can be found in each section.

- **Code examples by service (p. 3)** – A list of AWS services that contain examples of how to use each service with AWS SDKs. Use this section if you know which service you want to use.
- **Code examples by SDK (p. 2267)** – A list of AWS SDKs that contain examples of how to use AWS services with each SDK. Use this section if you know which SDK you want to use.

Within each section, the examples are divided into the following categories:

- *Actions* are code excerpts that show you how to call individual service functions.
- *Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.
- *Cross-service examples* are sample applications that work across multiple AWS services.

All of the examples in this library can also be found in the [AWS Code Examples GitHub repository](#). The GitHub repository also contains instructions on how to set up, run, and test the examples.

## Using AWS services with an AWS SDK

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.

SDK documentation	Code examples
<a href="#">AWS SDK for C++</a>	<a href="#">AWS SDK for C++ code examples</a>
<a href="#">AWS SDK for Go</a>	<a href="#">AWS SDK for Go code examples</a>
<a href="#">AWS SDK for Java</a>	<a href="#">AWS SDK for Java code examples</a>
<a href="#">AWS SDK for JavaScript</a>	<a href="#">AWS SDK for JavaScript code examples</a>
<a href="#">AWS SDK for Kotlin</a>	<a href="#">AWS SDK for Kotlin code examples</a>
<a href="#">AWS SDK for .NET</a>	<a href="#">AWS SDK for .NET code examples</a>
<a href="#">AWS SDK for PHP</a>	<a href="#">AWS SDK for PHP code examples</a>
<a href="#">AWS SDK for Python (Boto3)</a>	<a href="#">AWS SDK for Python (Boto3) code examples</a>
<a href="#">AWS SDK for Ruby</a>	<a href="#">AWS SDK for Ruby code examples</a>
<a href="#">AWS SDK for Rust</a>	<a href="#">AWS SDK for Rust code examples</a>
<a href="#">AWS SDK for Swift</a>	<a href="#">AWS SDK for Swift code examples</a>

**Example availability**

Can't find what you need? Request a code example by using the **Provide feedback** link at the bottom of this page.

# Code examples by service using AWS SDKs

The following code examples show how to use AWS services with an AWS software development kit (SDK).

## Code examples

- [Code examples for ACM using AWS SDKs \(p. 4\)](#)
- [Code examples for API Gateway using AWS SDKs \(p. 22\)](#)
- [Code examples for API Gateway Management API using AWS SDKs \(p. 49\)](#)
- [Code examples for Application Auto Scaling using AWS SDKs \(p. 50\)](#)
- [Code examples for Application Recovery Controller using AWS SDKs \(p. 51\)](#)
- [Code examples for Audit Manager using AWS SDKs \(p. 55\)](#)
- [Code examples for Aurora using AWS SDKs \(p. 63\)](#)
- [Code examples for AWS Batch using AWS SDKs \(p. 129\)](#)
- [Code examples for AWS CloudFormation using AWS SDKs \(p. 130\)](#)
- [Code examples for CloudFront using AWS SDKs \(p. 131\)](#)
- [Code examples for CloudWatch using AWS SDKs \(p. 135\)](#)
- [Code examples for CloudWatch Events using AWS SDKs \(p. 185\)](#)
- [Code examples for CloudWatch Logs using AWS SDKs \(p. 192\)](#)
- [Code examples for Amazon Cognito Identity using AWS SDKs \(p. 211\)](#)
- [Code examples for Amazon Cognito Identity Provider using AWS SDKs \(p. 215\)](#)
- [Code examples for Amazon Cognito Sync using AWS SDKs \(p. 274\)](#)
- [Code examples for Amazon Comprehend using AWS SDKs \(p. 276\)](#)
- [Code examples for AWS Config using AWS SDKs \(p. 316\)](#)
- [Code examples for Device Farm using AWS SDKs \(p. 318\)](#)
- [Code examples for DynamoDB using AWS SDKs \(p. 325\)](#)
- [Code examples for Amazon EBS using AWS SDKs \(p. 603\)](#)
- [Code examples for Amazon EC2 using AWS SDKs \(p. 605\)](#)
- [Code examples for Amazon EC2 Auto Scaling using AWS SDKs \(p. 684\)](#)
- [Code examples for Amazon ECR using AWS SDKs \(p. 739\)](#)
- [Code examples for Amazon ECS using AWS SDKs \(p. 740\)](#)
- [Code examples for Amazon EKS using AWS SDKs \(p. 742\)](#)
- [Code examples for Amazon EMR using AWS SDKs \(p. 744\)](#)
- [Code examples for EventBridge using AWS SDKs \(p. 753\)](#)
- [Code examples for AWS Glue using AWS SDKs \(p. 779\)](#)
- [Code examples for IAM using AWS SDKs \(p. 862\)](#)
- [Code examples for AWS IoT using AWS SDKs \(p. 1099\)](#)
- [Code examples for AWS KMS using AWS SDKs \(p. 1101\)](#)

- [Code examples for Amazon Keyspaces using AWS SDKs \(p. 1151\)](#)
- [Code examples for Kinesis using AWS SDKs \(p. 1171\)](#)
- [Code examples for Kinesis Data Analytics using AWS SDKs \(p. 1197\)](#)
- [Code examples for Lambda using AWS SDKs \(p. 1215\)](#)
- [Code examples for Amazon Lex using AWS SDKs \(p. 1289\)](#)
- [Code examples for Lookout for Vision using AWS SDKs \(p. 1290\)](#)
- [Code examples for MediaLive using AWS SDKs \(p. 1312\)](#)
- [Code examples for MediaPackage using AWS SDKs \(p. 1313\)](#)
- [Code examples for Organizations using AWS SDKs \(p. 1315\)](#)
- [Code examples for Amazon Personalize using AWS SDKs \(p. 1331\)](#)
- [Code examples for Amazon Personalize Events using AWS SDKs \(p. 1363\)](#)
- [Code examples for Amazon Personalize Runtime using AWS SDKs \(p. 1367\)](#)
- [Code examples for Amazon Pinpoint using AWS SDKs \(p. 1373\)](#)
- [Code examples for Amazon Pinpoint SMS and Voice API using AWS SDKs \(p. 1406\)](#)
- [Code examples for Amazon Polly using AWS SDKs \(p. 1410\)](#)
- [Code examples for QLDB using AWS SDKs \(p. 1429\)](#)
- [Code examples for Amazon RDS using AWS SDKs \(p. 1430\)](#)
- [Code examples for Amazon RDS Data Service using AWS SDKs \(p. 1505\)](#)
- [Code examples for Amazon Redshift using AWS SDKs \(p. 1509\)](#)
- [Code examples for Amazon Rekognition using AWS SDKs \(p. 1518\)](#)
- [Code examples for Route 53 using AWS SDKs \(p. 1612\)](#)
- [Code examples for Route 53 domain registration using AWS SDKs \(p. 1613\)](#)
- [Code examples for Amazon S3 using AWS SDKs \(p. 1632\)](#)
- [Code examples for S3 Glacier using AWS SDKs \(p. 1833\)](#)
- [Code examples for Amazon SES using AWS SDKs \(p. 1867\)](#)
- [Code examples for Amazon SES API v2 using AWS SDKs \(p. 1923\)](#)
- [Code examples for Amazon SNS using AWS SDKs \(p. 1928\)](#)
- [Code examples for Amazon SQS using AWS SDKs \(p. 2020\)](#)
- [Code examples for AWS STS using AWS SDKs \(p. 2090\)](#)
- [Code examples for SageMaker using AWS SDKs \(p. 2107\)](#)
- [Code examples for Secrets Manager using AWS SDKs \(p. 2122\)](#)
- [Code examples for Step Functions using AWS SDKs \(p. 2142\)](#)
- [Code examples for AWS Support using AWS SDKs \(p. 2157\)](#)
- [Code examples for Systems Manager using AWS SDKs \(p. 2200\)](#)
- [Code examples for Amazon Textract using AWS SDKs \(p. 2202\)](#)
- [Code examples for Amazon Transcribe using AWS SDKs \(p. 2220\)](#)
- [Code examples for Amazon Translate using AWS SDKs \(p. 2250\)](#)

## Code examples for ACM using AWS SDKs

The following code examples show how to use AWS Certificate Manager with an AWS software development kit (SDK).

### Code examples

- [Actions for ACM using AWS SDKs \(p. 5\)](#)
  - Add tags to an ACM certificate using an AWS SDK (p. 5)
  - Delete an ACM certificate using an AWS SDK (p. 6)
  - Describe an ACM certificate using an AWS SDK (p. 7)
  - Get an ACM certificate using an AWS SDK (p. 9)
  - Import an ACM certificate using an AWS SDK (p. 10)
  - List ACM certificates using an AWS SDK (p. 11)
  - List tags for an ACM certificate using an AWS SDK (p. 13)
  - Remove tags from an ACM certificate using an AWS SDK (p. 13)
  - Request validation of an ACM certificate using an AWS SDK (p. 14)
  - Resend validation email for an ACM certificate using an AWS SDK (p. 15)
- [Scenarios for ACM using AWS SDKs \(p. 16\)](#)
  - Manage ACM certificates using an AWS SDK (p. 16)

## Actions for ACM using AWS SDKs

The following code examples show how to use AWS Certificate Manager with AWS SDKs. Each example calls an individual service function.

### Examples

- [Add tags to an ACM certificate using an AWS SDK \(p. 5\)](#)
- [Delete an ACM certificate using an AWS SDK \(p. 6\)](#)
- [Describe an ACM certificate using an AWS SDK \(p. 7\)](#)
- [Get an ACM certificate using an AWS SDK \(p. 9\)](#)
- [Import an ACM certificate using an AWS SDK \(p. 10\)](#)
- [List ACM certificates using an AWS SDK \(p. 11\)](#)
- [List tags for an ACM certificate using an AWS SDK \(p. 13\)](#)
- [Remove tags from an ACM certificate using an AWS SDK \(p. 13\)](#)
- [Request validation of an ACM certificate using an AWS SDK \(p. 14\)](#)
- [Resend validation email for an ACM certificate using an AWS SDK \(p. 15\)](#)

## Add tags to an ACM certificate using an AWS SDK

The following code example shows how to add tags to ACM certificates.

### Python

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
    Encapsulates ACM functions.  
    """  
    def __init__(self, acm_client):
```

```
"""
:param acm_client: A Boto3 ACM client.
"""
self.acm_client = acm_client

def add_tags(self, certificate_arn, tags):
    """
    Adds tags to a certificate. Tags are key-value pairs that contain custom
    metadata.

    :param certificate_arn: The ARN of the certificate.
    :param tags: A dictionary of key-value tags to add to the certificate.
    """
    try:
        self.acm_client.add_tags_to_certificate(
            CertificateArn=certificate_arn,
            Tags=[{'Key': key, 'Value': value} for key, value in tags.items()])
        logger.info("Added %s tags to certificate %s.", len(tags),
                    certificate_arn)
    except ClientError:
        logger.exception("Couldn't add tags to certificate %s.",
                         certificate_arn)
        raise
```

- For API details, see [AddTagsToCertificate in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete an ACM certificate using an AWS SDK

The following code example shows how to delete ACM certificates.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """
    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client

    def remove(self, certificate_arn):
        """
        Removes a certificate.

        :param certificate_arn: The ARN of the certificate to remove.
        """
        try:
            self.acm_client.delete_certificate(CertificateArn=certificate_arn)
            logger.info("Removed certificate %s.", certificate_arn)
        except ClientError:
            logger.exception("Couldn't remove certificate %s.", certificate_arn)
```

```
raise
```

- For API details, see [DeleteCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe an ACM certificate using an AWS SDK

The following code examples show how to describe ACM certificates.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon;
using Amazon.CertificateManager;
using Amazon.CertificateManager.Model;
using System;
using System.Threading.Tasks;

namespace DescribeCertificate
{
    class DescribeCertificate
    {
        // The following example retrieves and displays the metadata for a
        // certificate using the AWS Certificate Manager (ACM) service. It
        // was created using AWS SDK for .NET 3.5 and .NET 5.0.

        // Specify your AWS Region (an example Region is shown).
        private static readonly RegionEndpoint ACMRegion = RegionEndpoint.USEast1;
        private static AmazonCertificateManagerClient _client;

        static void Main(string[] args)
        {
            var _client = new
                Amazon.CertificateManager.AmazonCertificateManagerClient(ACMRegion);

            var describeCertificateReq = new DescribeCertificateRequest();
            // The ARN used here is just an example. Replace it with the ARN of
            // a certificate that exists on your account.
            describeCertificateReq.CertificateArn =
                "arn:aws:acm:us-
east-1:123456789012:certificate/8cf7dae-9b6a-2d07-92bc-1c309EXAMPLE";

            var certificateDetailResp =
                DescribeCertificateResponseAsync(client: _client, request:
            describeCertificateReq);
            var certificateDetail = certificateDetailResp.Result.Certificate;

            if (certificateDetail is not null)
            {
                DisplayCertificateDetails(certificateDetail);
            }
        }

        /// <summary>
        /// Displays detailed metadata about a certificate retrieved
        /// using the ACM service.
        /// </summary>
```

```
    ///<param name="certificateDetail">The object that contains details
    /// returned from the call to DescribeCertificateAsync.</param>
    static void DisplayCertificateDetails(CertificateDetail certificateDetail)
    {
        Console.WriteLine("\nCertificate Details: ");
        Console.WriteLine($"Certificate Domain:
{certificateDetail.DomainName}");
        Console.WriteLine($"Certificate Arn:
{certificateDetail.CertificateArn}");
        Console.WriteLine($"Certificate Subject: {certificateDetail.Subject}");
        Console.WriteLine($"Certificate Status: {certificateDetail.Status}");
        foreach (var san in certificateDetail.SubjectAlternativeNames)
        {
            Console.WriteLine($"Certificate SubjectAlternativeName: {san}");
        }
    }

    ///<summary>
    /// Retrieves the metadata associated with the ACM service certificate.
    ///</summary>
    ///<param name="client">An AmazonCertificateManagerClient object
    /// used to call DescribeCertificateResponse.</param>
    ///<param name="request">The DescribeCertificateRequest object that
    /// will be passed to the method call.</param>
    ///<returns></returns>
    static async Task<DescribeCertificateResponse>
DescribeCertificateResponseAsync(
    AmazonCertificateManagerClient client, DescribeCertificateRequest
request)
{
    var response = new DescribeCertificateResponse();

    try
    {
        response = await client.DescribeCertificateAsync(request);
    }
    catch (InvalidArnException ex)
    {
        Console.WriteLine($"Error: The ARN specified is invalid.");
    }
    catch (ResourceNotFoundException ex)
    {
        Console.WriteLine($"Error: The specified certificate could not be
found.");
    }

    return response;
}
}
```

- For API details, see [DescribeCertificate](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
    Encapsulates ACM functions.  
    """  
    def __init__(self, acm_client):  
        """  
        :param acm_client: A Boto3 ACM client.  
        """  
        self.acm_client = acm_client  
  
    def describe(self, certificate_arn):  
        """  
        Gets certificate metadata.  
  
        :param certificate_arn: The Amazon Resource Name (ARN) of the certificate.  
        :return: Metadata about the certificate.  
        """  
        try:  
            response = self.acm_client.describe_certificate(  
                CertificateArn=certificate_arn)  
            certificate = response['Certificate']  
            logger.info(  
                "Got metadata for certificate for domain %s.",  
                certificate['DomainName'])  
        except ClientError:  
            logger.exception("Couldn't get data for certificate %s.",  
                             certificate_arn)  
            raise  
        else:  
            return certificate
```

- For API details, see [DescribeCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get an ACM certificate using an AWS SDK

The following code example shows how to get ACM certificates.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
    Encapsulates ACM functions.  
    """  
    def __init__(self, acm_client):  
        """  
        :param acm_client: A Boto3 ACM client.  
        """  
        self.acm_client = acm_client  
  
    def get(self, certificate_arn):  
        """  
        Gets the body and certificate chain of a certificate.  
        """
```

```
:param certificate_arn: The ARN of the certificate.  
:return: The body and chain of a certificate.  
"""  
try:  
    response =  
self.acm_client.get_certificate(CertificateArn=certificate_arn)  
    logger.info("Got certificate %s and its chain.", certificate_arn)  
except ClientError:  
    logger.exception("Couldn't get certificate %s.", certificate_arn)  
    raise  
else:  
    return response
```

- For API details, see [GetCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Import an ACM certificate using an AWS SDK

The following code example shows how to import ACM certificates.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
    Encapsulates ACM functions.  
    """  
    def __init__(self, acm_client):  
        """  
        :param acm_client: A Boto3 ACM client.  
        """  
        self.acm_client = acm_client  
  
    def import_certificate(self, certificate_body, private_key):  
        """  
        Imports a self-signed certificate to ACM.  
  
        :param certificate_body: The body of the certificate, in PEM format.  
        :param private_key: The unencrypted private key of the certificate, in PEM  
                           format.  
        :return: The ARN of the imported certificate.  
        """  
        try:  
            response = self.acm_client.import_certificate(  
                Certificate=certificate_body, PrivateKey=private_key)  
            certificate_arn = response['CertificateArn']  
            logger.info("Imported certificate.")  
        except ClientError:  
            logger.exception("Couldn't import certificate.")  
            raise  
        else:  
            return certificate_arn
```

- For API details, see [ImportCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

## List ACM certificates using an AWS SDK

The following code examples show how to list ACM certificates.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon;
using Amazon.CertificateManager;
using Amazon.CertificateManager.Model;
using System;
using System.Threading.Tasks;

namespace ListCertificates
{
    // The following example retrieves and displays a list of the
    // certificates defined for the default account using the AWS
    // Certificate Manager (ACM) service. It was created using
    // AWS SDK for .NET 3.5 and .NET 5.0.
    class ListCertificates
    {
        // Specify your AWS Region (an example Region is shown).

        private static readonly RegionEndpoint ACMRegion = RegionEndpoint.USEast1;
        private static AmazonCertificateManagerClient _client;

        static void Main(string[] args)
        {
            var _client = new AmazonCertificateManagerClient(ACMRegion);
            var certificateList = ListCertificatesResponseAsync(client: _client);

            Console.WriteLine("Certificate Summary List\n");

            foreach (var certificate in
certificateList.Result.CertificateSummaryList)
            {
                Console.WriteLine($"Certificate Domain: {certificate.DomainName}");
                Console.WriteLine($"Certificate ARN:
{certificate.CertificateArn}\n");
            }
        }

        /// <summary>
        /// Retrieves a list of the certificates defined in this Region.
        /// </summary>
        /// <param name="client">The ACM client object passed to the
        /// ListCertificatesResAsync method call.</param>
        /// <param name="request"></param>
        /// <returns>The ListCertificatesResponse.</returns>
        static async Task<ListCertificatesResponse> ListCertificatesResponseAsync(
            AmazonCertificateManagerClient client)
        {
            var request = new ListCertificatesRequest();

            var response = await client.ListCertificatesAsync(request);
            return response;
        }
    }
}
```

```
        }  
    }  
}
```

- For API details, see [ListCertificates](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
    Encapsulates ACM functions.  
    """  
    def __init__(self, acm_client):  
        """  
        :param acm_client: A Boto3 ACM client.  
        """  
        self.acm_client = acm_client  
  
    def list(  
            self, max_items, statuses=None, key_usage=None,  
            extended_key_usage=None,  
            key_types=None):  
        """  
        Lists the certificates for the current account.  
  
        :param max_items: The maximum number of certificates to list.  
        :param statuses: Filters the results to the specified statuses. If None,  
        all  
                certificates are included.  
        :param key_usage: Filters the results to the specified key usages. If None,  
                all key usages are included.  
        :param extended_key_usage: Filters the results to the specified extended  
        key  
                usages. If None, all extended key usages are included.  
        :param key_types: Filters the results to the specified key types. If None,  
        all  
                key types are included.  
        :return: The list of certificates.  
        """  
        try:  
            kwargs = {'MaxItems': max_items}  
            if statuses is not None:  
                kwargs['CertificateStatuses'] = statuses  
            includes = {}  
            if key_usage is not None:  
                includes['keyUsage'] = key_usage  
            if extended_key_usage is not None:  
                includes['extendedKeyUsage'] = extended_key_usage  
            if key_types is not None:  
                includes['keyTypes'] = key_types  
            if includes:  
                kwargs['Includes'] = includes  
            response = self.acm_client.list_certificates(**kwargs)
```

```
certificates = response['CertificateSummaryList']
logger.info("Got %s certificates.", len(certificates))
except ClientError:
    logger.exception("Couldn't get certificates.")
    raise
else:
    return certificates
```

- For API details, see [ListCertificates](#) in *AWS SDK for Python (Boto3) API Reference*.

## List tags for an ACM certificate using an AWS SDK

The following code example shows how to list tags for ACM certificates.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """
    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client

    def list_tags(self, certificate_arn):
        """
        Lists the tags attached to a certificate.

        :param certificate_arn: The ARN of the certificate.
        :return: The dictionary of certificate tags.
        """
        try:
            response = self.acm_client.list_tags_for_certificate(
                CertificateArn=certificate_arn)
            tags = {tag['Key']: tag['Value'] for tag in response['Tags']}
            logger.info("Got %s tags for certificates %s.", len(tags),
                        certificate_arn)
        except ClientError:
            logger.exception("Couldn't get tags for certificate %s.",
                            certificate_arn)
            raise
        else:
            return tags
```

- For API details, see [ListTagsForCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Remove tags from an ACM certificate using an AWS SDK

The following code example shows how to remove tags from ACM certificates.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
    Encapsulates ACM functions.  
    """  
    def __init__(self, acm_client):  
        """  
        :param acm_client: A Boto3 ACM client.  
        """  
        self.acm_client = acm_client  
  
    def remove_tags(self, certificate_arn, tags):  
        """  
        Removes tags from a certificate. If the value of a tag is specified, the  
        tag is removed only when the value matches the value of the certificate's tag.  
        Otherwise, the tag is removed regardless of its value.  
        :param certificate_arn: The ARN of the certificate.  
        :param tags: The dictionary of tags to remove.  
        """  
        try:  
            cert_tags = []  
            for key, value in tags.items():  
                tag = {'Key': key}  
                if value is not None:  
                    tag['Value'] = value  
                cert_tags.append(tag)  
            self.acm_client.remove_tags_from_certificate(  
                CertificateArn=certificate_arn, Tags=cert_tags)  
            logger.info(  
                "Removed %s tags from certificate %s.", len(tags), certificate_arn)  
        except ClientError:  
            logger.exception(  
                "Couldn't remove tags from certificate %s.", certificate_arn)  
            raise
```

- For API details, see [RemoveTagsFromCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Request validation of an ACM certificate using an AWS SDK

The following code example shows how to request validation of ACM certificates.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:
```

```
"""
Encapsulates ACM functions.
"""
def __init__(self, acm_client):
    """
    :param acm_client: A Boto3 ACM client.
    """
    self.acm_client = acm_client

    def request_validation(
        self, domain, alternate_domains, method, validation_domains=None):
        """
        Starts a validation request that results in a new certificate being issued by ACM. DNS validation requires that you add CNAME records to your DNS provider. Email validation sends email to a list of email addresses that are associated with the domain.

        For more information, see \_Issuing and managing certificates\_ in the ACM user guide.
        https://docs.aws.amazon.com/acm/latest/userguide/gs.html

        :param domain: The primary domain to associate with the certificate.
        :param alternate_domains: Subject Alternate Names (SANs) for the certificate.
        :param method: The validation method, either DNS or EMAIL.
        :param validation_domains: Alternate domains to use for email validation,
    when
                                the email domain differs from the primary domain
    of
                                the certificate.
    :return: The ARN of the requested certificate.
    """
    try:
        kwargs = {
            'DomainName': domain,
            'ValidationMethod': method,
            'SubjectAlternativeNames': alternate_domains}
        if validation_domains is not None:
            kwargs['DomainValidationOptions'] = [{{
                'DomainName': key,
                'ValidationDomain': value
            }} for key, value in validation_domains.items()]
        response = self.acm_client.request_certificate(**kwargs)
        certificate_arn = response['CertificateArn']
        logger.info(
            "Requested %s validation for domain %s. Certificate ARN is %s.",
            method, domain, certificate_arn)
    except ClientError:
        logger.exception(
            "Request for %s validation of domain %s failed.", method, domain)
        raise
    else:
        return certificate_arn
    
```

- For API details, see [RequestCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Resend validation email for an ACM certificate using an AWS SDK

The following code example shows how to resend validation email for ACM certificates.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
    Encapsulates ACM functions.  
    """  
    def __init__(self, acm_client):  
        """  
        :param acm_client: A Boto3 ACM client.  
        """  
        self.acm_client = acm_client  
  
    def resend_validation_email(self, certificate_arn, domain, validation_domain):  
        """  
        Request that validation email is sent again, for a certificate that was  
        previously requested with email validation.  
  
        :param certificate_arn: The ARN of the certificate.  
        :param domain: The primary domain of the certificate.  
        :param validation_domain: Alternate domain to use for determining email  
            addresses to use for validation.  
        """  
        try:  
            self.acm_client.resend_validation_email(  
                CertificateArn=certificate_arn,  
                Domain=domain,  
                ValidationDomain=validation_domain)  
            logger.info(  
                "Validation email resent to validation domain %s.",  
                validation_domain)  
        except ClientError:  
            logger.exception(  
                "Couldn't resend validation email to %s.", validation_domain)  
            raise
```

- For API details, see [ResendValidationEmail](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios for ACM using AWS SDKs

The following code examples show how to use AWS Certificate Manager with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

#### Examples

- [Manage ACM certificates using an AWS SDK \(p. 16\)](#)

## Manage ACM certificates using an AWS SDK

The following code example shows how to:

- Request a certificate from ACM.
- Import a self-signed certificate.

- List and describe certificates.
- Remove certificates.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps ACM operations.

```
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class AcmCertificate:
    """
    Encapsulates ACM functions.
    """
    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client

    def request_validation(
            self, domain, alternate_domains, method, validation_domains=None):
        """
        Starts a validation request that results in a new certificate being issued
        by ACM. DNS validation requires that you add CNAME records to your DNS
        provider. Email validation sends email to a list of email addresses that
        are associated with the domain.

        For more information, see _Issuing and managing certificates_ in the ACM
        user guide.
        https://docs.aws.amazon.com/acm/latest/userguide/gs.html

        :param domain: The primary domain to associate with the certificate.
        :param alternate_domains: Subject Alternative Names (SANs) for the
        certificate.
        :param method: The validation method, either DNS or EMAIL.
        :param validation_domains: Alternate domains to use for email validation,
        when
                                         the email domain differs from the primary domain
        of
                                         the certificate.
        :return: The ARN of the requested certificate.
        """
        try:
            kwargs = {
                'DomainName': domain,
                'ValidationMethod': method,
                'SubjectAlternativeNames': alternate_domains}
            if validation_domains is not None:
                kwargs['DomainValidationOptions'] = [{}
```

```
        'DomainName': key,
        'ValidationDomain': value
    } for key, value in validation_domains.items()]
response = self.acm_client.request_certificate(**kwargs)
certificate_arn = response['CertificateArn']
logger.info(
    "Requested %s validation for domain %s. Certificate ARN is %s.",
    method, domain, certificate_arn)
except ClientError:
    logger.exception(
        "Request for %s validation of domain %s failed.", method, domain)
    raise
else:
    return certificate_arn

def import_certificate(self, certificate_body, private_key):
    """
    Imports a self-signed certificate to ACM.

    :param certificate_body: The body of the certificate, in PEM format.
    :param private_key: The unencrypted private key of the certificate, in PEM
                        format.
    :return: The ARN of the imported certificate.
    """
    try:
        response = self.acm_client.import_certificate(
            Certificate=certificate_body, PrivateKey=private_key)
        certificate_arn = response['CertificateArn']
        logger.info("Imported certificate.")
    except ClientError:
        logger.exception("Couldn't import certificate.")
        raise
    else:
        return certificate_arn

def list(
    self, max_items, statuses=None, key_usage=None,
extended_key_usage=None,
key_types=None):
    """
    Lists the certificates for the current account.

    :param max_items: The maximum number of certificates to list.
    :param statuses: Filters the results to the specified statuses. If None,
all
                    certificates are included.
    :param key_usage: Filters the results to the specified key usages. If None,
all key usages are included.
    :param extended_key_usage: Filters the results to the specified extended
key
                                usages. If None, all extended key usages are
                                included.
    :param key_types: Filters the results to the specified key types. If None,
all
                    key types are included.
    :return: The list of certificates.
    """
    try:
        kwargs = {'MaxItems': max_items}
        if statuses is not None:
            kwargs['CertificateStatuses'] = statuses
        includes = {}
        if key_usage is not None:
            includes['keyUsage'] = key_usage
        if extended_key_usage is not None:
            includes['extendedKeyUsage'] = extended_key_usage
```

```
if key_types is not None:
    includes['keyTypes'] = key_types
if includes:
    kwargs['Includes'] = includes
response = self.acm_client.list_certificates(**kwargs)
certificates = response['CertificateSummaryList']
logger.info("Got %s certificates.", len(certificates))
except ClientError:
    logger.exception("Couldn't get certificates.")
    raise
else:
    return certificates

def describe(self, certificate_arn):
    """
    Gets certificate metadata.

    :param certificate_arn: The Amazon Resource Name (ARN) of the certificate.
    :return: Metadata about the certificate.
    """
    try:
        response = self.acm_client.describe_certificate(
            CertificateArn=certificate_arn)
        certificate = response['Certificate']
        logger.info(
            "Got metadata for certificate for domain %s.",
            certificate['DomainName'])
    except ClientError:
        logger.exception("Couldn't get data for certificate %s.",
                         certificate_arn)
        raise
    else:
        return certificate

def get(self, certificate_arn):
    """
    Gets the body and certificate chain of a certificate.

    :param certificate_arn: The ARN of the certificate.
    :return: The body and chain of a certificate.
    """
    try:
        response =
self.acm_client.get_certificate(CertificateArn=certificate_arn)
        logger.info("Got certificate %s and its chain.", certificate_arn)
    except ClientError:
        logger.exception("Couldn't get certificate %s.", certificate_arn)
        raise
    else:
        return response

def add_tags(self, certificate_arn, tags):
    """
    Adds tags to a certificate. Tags are key-value pairs that contain custom
    metadata.

    :param certificate_arn: The ARN of the certificate.
    :param tags: A dictionary of key-value tags to add to the certificate.
    """
    try:
        self.acm_client.add_tags_to_certificate(
            CertificateArn=certificate_arn,
            Tags=[{'Key': key, 'Value': value} for key, value in tags.items()])
        logger.info("Added %s tags to certificate %s.", len(tags),
                    certificate_arn)
    except ClientError:
```

```
logger.exception("Couldn't add tags to certificate %s.",  
certificate_arn)  
raise  
  
def list_tags(self, certificate_arn):  
    """  
    Lists the tags attached to a certificate.  
  
    :param certificate_arn: The ARN of the certificate.  
    :return: The dictionary of certificate tags.  
    """  
    try:  
        response = self.acm_client.list_tags_for_certificate(  
            CertificateArn=certificate_arn)  
        tags = {tag['Key']: tag['Value'] for tag in response['Tags']}  
        logger.info("Got %s tags for certificates %s.", len(tags),  
certificate_arn)  
    except ClientError:  
        logger.exception("Couldn't get tags for certificate %s.",  
certificate_arn)  
        raise  
    else:  
        return tags  
  
def remove_tags(self, certificate_arn, tags):  
    """  
    Removes tags from a certificate. If the value of a tag is specified, the  
tag is  
removed only when the value matches the value of the certificate's tag.  
Otherwise, the tag is removed regardless of its value.  
  
    :param certificate_arn: The ARN of the certificate.  
    :param tags: The dictionary of tags to remove.  
    """  
    try:  
        cert_tags = []  
        for key, value in tags.items():  
            tag = {'Key': key}  
            if value is not None:  
                tag['Value'] = value  
            cert_tags.append(tag)  
        self.acm_client.remove_tags_from_certificate(  
            CertificateArn=certificate_arn, Tags=cert_tags)  
        logger.info(  
            "Removed %s tags from certificate %s.", len(tags), certificate_arn)  
    except ClientError:  
        logger.exception(  
            "Couldn't remove tags from certificate %s.", certificate_arn)  
        raise  
  
def remove(self, certificate_arn):  
    """  
    Removes a certificate.  
  
    :param certificate_arn: The ARN of the certificate to remove.  
    """  
    try:  
        self.acm_client.delete_certificate(CertificateArn=certificate_arn)  
        logger.info("Removed certificate %s.", certificate_arn)  
    except ClientError:  
        logger.exception("Couldn't remove certificate %s.", certificate_arn)  
        raise
```

Use the wrapper class to manage certificates for your account.

```
def usage_demo():
    print('*'*88)
    print("Welcome to the AWS Certificate Manager (ACM) demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    acm_certificate = AcmCertificate(boto3.client('acm'))
    domain = 'example.com'
    sub_domains = [f'{sub}.{domain}' for sub in ['test', 'dev']]
    print(f"Request a certificate for {domain}.")
    certificate_arn = acm_certificate.request_validation(domain, sub_domains,
'DNS')
    print(f"Started validation, got certificate ARN: {certificate_arn}.")

    import_cert_arn = None
    cert_file_name = input(
        "Enter the file name for a self-signed certificate in PEM format. "
        "This certificate will be imported to ACM. Press Enter to skip: ")
    if cert_file_name:
        pk_file_name = input(
            "Enter the file name for the unencrypted private key of the
certificate. "
            "This file must also be in PEM format: ")
        if pk_file_name:
            with open(cert_file_name, 'rb') as cert_file:
                import_cert = cert_file.read()
            with open(pk_file_name, 'rb') as pk_file:
                import_pk = pk_file.read()
            import_cert_arn = acm_certificate.import_certificate(import_cert,
import_pk)
            print(f"Certificate imported, got ARN: {import_cert_arn}")
        else:
            print("No private key file entered. Skipping certificate import.")
    else:
        print("Skipping self-signed certificate import.")

    print("Getting the first 10 issued certificates.")
    certificates = acm_certificate.list(10, statuses=['ISSUED'])
    print(f"Found {len(certificates)} issued certificates.")

    print(f"Getting metadata for certificate {certificate_arn}")
    cert_metadata = acm_certificate.describe(certificate_arn)
    pprint(cert_metadata)

    if import_cert_arn is not None:
        print(f"Getting certificate for imported certificate {import_cert_arn}")
        import_cert_data = acm_certificate.get(import_cert_arn)
        pprint(import_cert_data)

    print(f"Adding tags to certificate {certificate_arn}.")
    acm_certificate.add_tags(certificate_arn, {
        'purpose': 'acm demo',
        'color': 'green'})
    tags = acm_certificate.list_tags(certificate_arn)
    print(f"Found tags: {tags}")
    acm_certificate.remove_tags(certificate_arn, {key: None for key in tags})
    print("Removed tags.")

    print("Removing certificates added during the demo.")
    acm_certificate.remove(certificate_arn)
    if import_cert_arn is not None:
        acm_certificate.remove(import_cert_arn)

    print("Thanks for watching!")
```

```
print('-'*88)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [AddTagsToCertificate](#)
  - [DeleteCertificate](#)
  - [DescribeCertificate](#)
  - [GetCertificate](#)
  - [ImportCertificate](#)
  - [ListCertificates](#)
  - [ListTagsForCertificate](#)
  - [RemoveTagsFromCertificate](#)
  - [RequestCertificate](#)
  - [ResendValidationEmail](#)

## Code examples for API Gateway using AWS SDKs

The following code examples show how to use Amazon API Gateway with an AWS software development kit (SDK).

### Code examples

- [Actions for API Gateway using AWS SDKs \(p. 23\)](#)
  - [Add a response from an AWS service integrated with an API Gateway method using an AWS SDK \(p. 23\)](#)
  - [Add an HTTP method to an API Gateway REST resource using an AWS SDK \(p. 25\)](#)
  - [Add an HTTP response to an API Gateway REST resource using an AWS SDK \(p. 26\)](#)
  - [Create an API Gateway REST API using an AWS SDK \(p. 28\)](#)
  - [Create an API Gateway REST resource using an AWS SDK \(p. 30\)](#)
  - [Delete an API Gateway REST API using an AWS SDK \(p. 30\)](#)
  - [Delete an API Gateway deployment using an AWS SDK \(p. 32\)](#)
  - [Deploy an API Gateway REST API using an AWS SDK \(p. 32\)](#)
  - [Get API Gateway REST resources using an AWS SDK \(p. 34\)](#)
  - [Get the base path mapping for a custom domain name in API Gateway \(p. 35\)](#)
  - [Integrate an API Gateway method with an AWS service using an AWS SDK \(p. 36\)](#)
  - [List API Gateway REST APIs using an AWS SDK \(p. 37\)](#)
  - [List the base path mapping for a custom domain name in API Gateway \(p. 39\)](#)
  - [Update the base path mapping for a custom domain name in API Gateway \(p. 40\)](#)
- [Scenarios for API Gateway using AWS SDKs \(p. 41\)](#)
  - [Create and deploy a REST API using an AWS SDK \(p. 41\)](#)
- [Cross-service examples for API Gateway using AWS SDKs \(p. 46\)](#)
  - [Create an API Gateway REST API to track COVID-19 data \(p. 46\)](#)
  - [Create a lending library REST API \(p. 47\)](#)
  - [Create a websocket chat application with API Gateway \(p. 47\)](#)
- [Use API Gateway to invoke a Lambda function \(p. 48\)](#)

## Actions for API Gateway using AWS SDKs

The following code examples show how to use Amazon API Gateway with AWS SDKs. Each example calls an individual service function.

### Examples

- [Add a response from an AWS service integrated with an API Gateway method using an AWS SDK \(p. 23\)](#)
- [Add an HTTP method to an API Gateway REST resource using an AWS SDK \(p. 25\)](#)
- [Add an HTTP response to an API Gateway REST resource using an AWS SDK \(p. 26\)](#)
- [Create an API Gateway REST API using an AWS SDK \(p. 28\)](#)
- [Create an API Gateway REST resource using an AWS SDK \(p. 30\)](#)
- [Delete an API Gateway REST API using an AWS SDK \(p. 30\)](#)
- [Delete an API Gateway deployment using an AWS SDK \(p. 32\)](#)
- [Deploy an API Gateway REST API using an AWS SDK \(p. 32\)](#)
- [Get API Gateway REST resources using an AWS SDK \(p. 34\)](#)
- [Get the base path mapping for a custom domain name in API Gateway \(p. 35\)](#)
- [Integrate an API Gateway method with an AWS service using an AWS SDK \(p. 36\)](#)
- [List API Gateway REST APIs using an AWS SDK \(p. 37\)](#)
- [List the base path mapping for a custom domain name in API Gateway \(p. 39\)](#)
- [Update the base path mapping for a custom domain name in API Gateway \(p. 40\)](#)

## Add a response from an AWS service integrated with an API Gateway method using an AWS SDK

The following code example shows how to add a response from an AWS service integrated with an API Gateway method.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:  
    """  
        Encapsulates Amazon API Gateway functions that are used to create a REST API  
        that  
        integrates with another AWS service.  
    """  
    def __init__(self, apig_client):  
        """  
            :param apig_client: A Boto3 API Gateway client.  
        """  
        self.apig_client = apig_client  
        self.api_id = None  
        self.root_id = None  
        self.stage = None
```

```

def add_integration_method(
    self, resource_id, rest_method, service_endpoint_prefix,
service_action,
        service_method, role_arn, mapping_template):
    """
    Adds an integration method to a REST API. An integration method is a REST
    resource, such as '/users', and an HTTP verb, such as GET. The integration
    method is backed by an AWS service, such as Amazon DynamoDB.

    :param resource_id: The ID of the REST resource.
    :param rest_method: The HTTP verb used with the REST resource.
    :param service_endpoint_prefix: The service endpoint that is integrated
with
                this method, such as 'dynamodb'.
    :param service_action: The action that is called on the service, such as
'GetItem'.
    :param service_method: The HTTP method of the service request, such as
POST.
    :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
Gateway permission to use the specified action with the
service.
    :param mapping_template: A mapping template that is used to translate REST
elements, such as query parameters, to the request
body format required by the service.
    """
    service_uri = (f'arn:aws:apigateway:{self.apig_client.meta.region_name}'
                  f':{service_endpoint_prefix}:action/{service_action}')
    try:
        self.apig_client.put_method(
            restApiId=self.api_id,
            resourceId=resource_id,
            httpMethod=rest_method,
            authorizationType='NONE')
        self.apig_client.put_method_response(
            restApiId=self.api_id,
            resourceId=resource_id,
            httpMethod=rest_method,
            statusCode='200',
            responseModels={'application/json': 'Empty'})
        logger.info("Created %s method for resource %s.", rest_method,
resource_id)
    except ClientError:
        logger.exception(
            "Couldn't create %s method for resource %s.", rest_method,
resource_id)
        raise

    try:
        self.apig_client.put_integration(
            restApiId=self.api_id,
            resourceId=resource_id,
            httpMethod=rest_method,
            type='AWS',
            integrationHttpMethod=service_method,
            credentials=role_arn,
            requestTemplates={'application/json':
json.dumps(mapping_template)},
            uri=service_uri,
            passthroughBehavior='WHEN_NO_TEMPLATES')
        self.apig_client.put_integration_response(
            restApiId=self.api_id,
            resourceId=resource_id,
            httpMethod=rest_method,
            statusCode='200',
            responseTemplates={'application/json': ''})
        logger.info(

```

```
        "Created integration for resource %s to service URI %s.",  
        resource_id,  
        service_uri)  
    except ClientError:  
        logger.exception(  
            "Couldn't create integration for resource %s to service URI %s.",  
            resource_id, service_uri)  
    raise
```

- For API details, see [PutIntegrationResponse](#) in *AWS SDK for Python (Boto3) API Reference*.

## Add an HTTP method to an API Gateway REST resource using an AWS SDK

The following code example shows how to add an HTTP method to an API Gateway REST resource.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:  
    """  
        Encapsulates Amazon API Gateway functions that are used to create a REST API  
        that  
        integrates with another AWS service.  
    """  
    def __init__(self, apig_client):  
        """  
            :param apig_client: A Boto3 API Gateway client.  
        """  
        self.apig_client = apig_client  
        self.api_id = None  
        self.root_id = None  
        self.stage = None  
  
    def add_integration_method(  
        self, resource_id, rest_method, service_endpoint_prefix,  
        service_action,  
        service_method, role_arn, mapping_template):  
        """  
            Adds an integration method to a REST API. An integration method is a REST  
            resource, such as '/users', and an HTTP verb, such as GET. The integration  
            method is backed by an AWS service, such as Amazon DynamoDB.  
  
            :param resource_id: The ID of the REST resource.  
            :param rest_method: The HTTP verb used with the REST resource.  
            :param service_endpoint_prefix: The service endpoint that is integrated  
                with  
                this method, such as 'dynamodb'.  
            :param service_action: The action that is called on the service, such as  
                'GetItem'.  
            :param service_method: The HTTP method of the service request, such as  
                POST.  
            :param role_arn: The Amazon Resource Name (ARN) of a role that grants API  
                Gateway permission to use the specified action with the
```

```
        service.
:param mapping_template: A mapping template that is used to translate REST
                           elements, such as query parameters, to the request
                           body format required by the service.
"""
service_uri = (f'arn:aws:apigateway:{self.apig_client.meta.region_name}'
               f'{service_endpoint_prefix}:action/{service_action}')
try:
    self.apig_client.put_method(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        authorizationType='NONE')
    self.apig_client.put_method_response(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        statusCode='200',
        responseModels={'application/json': 'Empty'})
    logger.info("Created %s method for resource %s.", rest_method,
resource_id)
except ClientError:
    logger.exception(
        "Couldn't create %s method for resource %s.", rest_method,
resource_id)
    raise

try:
    self.apig_client.put_integration(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        type='AWS',
        integrationHttpMethod=service_method,
        credentials=role_arn,
        requestTemplates={'application/json':
json.dumps(mapping_template)},
        uri=service_uri,
        passthroughBehavior='WHEN_NO_TEMPLATES')
    self.apig_client.put_integration_response(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        statusCode='200',
        responseTemplates={'application/json': ''})
    logger.info(
        "Created integration for resource %s to service URI %s.",
resource_id,
        service_uri)
except ClientError:
    logger.exception(
        "Couldn't create integration for resource %s to service URI %s.",
resource_id, service_uri)
    raise
```

- For API details, see [PutMethod in AWS SDK for Python \(Boto3\) API Reference](#).

## Add an HTTP response to an API Gateway REST resource using an AWS SDK

The following code example shows how to add an HTTP response to an API Gateway REST resource.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:  
    """  
        Encapsulates Amazon API Gateway functions that are used to create a REST API  
        that  
        integrates with another AWS service.  
    """  
    def __init__(self, apig_client):  
        """  
            :param apig_client: A Boto3 API Gateway client.  
        """  
        self.apig_client = apig_client  
        self.api_id = None  
        self.root_id = None  
        self.stage = None  
  
    def add_integration_method(  
        self, resource_id, rest_method, service_endpoint_prefix,  
        service_action,  
        service_method, role_arn, mapping_template):  
        """  
            Adds an integration method to a REST API. An integration method is a REST  
            resource, such as '/users', and an HTTP verb, such as GET. The integration  
            method is backed by an AWS service, such as Amazon DynamoDB.  
  
            :param resource_id: The ID of the REST resource.  
            :param rest_method: The HTTP verb used with the REST resource.  
            :param service_endpoint_prefix: The service endpoint that is integrated  
            with  
                this method, such as 'dynamodb'.  
            :param service_action: The action that is called on the service, such as  
                'GetItem'.  
            :param service_method: The HTTP method of the service request, such as  
            POST.  
            :param role_arn: The Amazon Resource Name (ARN) of a role that grants API  
                Gateway permission to use the specified action with the  
                service.  
            :param mapping_template: A mapping template that is used to translate REST  
                elements, such as query parameters, to the request  
                body format required by the service.  
        """  
        service_uri = (f'arn:aws:apigateway:{self.apig_client.meta.region_name}'  
                      f':{service_endpoint_prefix}:action/{service_action}')  
        try:  
            self.apig_client.put_method(  
                restApiId=self.api_id,  
                resourceId=resource_id,  
                httpMethod=rest_method,  
                authorizationType='NONE')  
            self.apig_client.put_method_response(  
                restApiId=self.api_id,  
                resourceId=resource_id,  
                httpMethod=rest_method,  
                statusCode='200',  
                responseModels={'application/json': 'Empty'})  
            logger.info("Created %s method for resource %s.", rest_method,  
                       resource_id)
```

```
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method,
resource_id)
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type='AWS',
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={'application/json':
json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior='WHEN_NO_TEMPLATES')
            self.apig_client.put_integration_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode='200',
                responseTemplates={'application/json': ''})
            logger.info(
                "Created integration for resource %s to service URI %s.",
resource_id,
                service_uri)
        except ClientError:
            logger.exception(
                "Couldn't create integration for resource %s to service URI %s.",
                resource_id, service_uri)
            raise
```

- For API details, see [PutMethodResponse](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an API Gateway REST API using an AWS SDK

The following code examples show how to create an API Gateway REST API.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createAPI( ApiGatewayClient apiGateway, String restApiId,
String restApiName) {

    try {
        CreateRestApiRequest request = CreateRestApiRequest.builder()
            .cloneFrom(restApiId)
            .description("Created using the Gateway Java API")
            .name(restApiName)
            .build();

        CreateRestApiResponse response = apiGateway.createRestApi(request);
        System.out.println("The id of the new api is "+response.id());
```

```
        return response.id();

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateRestApi](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
    that
        integrates with another AWS service.
    """
    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None

    def create_rest_api(self, api_name):
        """
        Creates a REST API on API Gateway. The default API has only a root resource
        and no HTTP methods.

        :param api_name: The name of the API. This descriptive name is not used in
            the API path.
        :return: The ID of the newly created API.
        """
        try:
            result = self.apig_client.create_rest_api(name=api_name)
            self.api_id = result['id']
            logger.info("Created REST API %s with ID %s.", api_name, self.api_id)
        except ClientError:
            logger.exception("Couldn't create REST API %s.", api_name)
            raise

        try:
            result = self.apig_client.get_resources(restApiId=self.api_id)
            self.root_id = next(
                item for item in result['items'] if item['path'] == '/')['id']
        except ClientError:
            logger.exception("Couldn't get resources for API %s.", self.api_id)
            raise
        except StopIteration as err:
            logger.exception("No root resource found in API %s.", self.api_id)
            raise ValueError from err
```

```
    return self.api_id
```

- For API details, see [CreateRestApi](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an API Gateway REST resource using an AWS SDK

The following code example shows how to create an API Gateway REST resource.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:  
    """  
        Encapsulates Amazon API Gateway functions that are used to create a REST API  
        that  
        integrates with another AWS service.  
    """  
    def __init__(self, apig_client):  
        """  
            :param apig_client: A Boto3 API Gateway client.  
        """  
        self.apig_client = apig_client  
        self.api_id = None  
        self.root_id = None  
        self.stage = None  
  
    def add_rest_resource(self, parent_id, resource_path):  
        """  
            Adds a resource to a REST API.  
  
            :param parent_id: The ID of the parent resource.  
            :param resource_path: The path of the new resource, relative to the parent.  
            :return: The ID of the new resource.  
        """  
        try:  
            result = self.apig_client.create_resource(  
                restApiId=self.api_id, parentId=parent_id, pathPart=resource_path)  
            resource_id = result['id']  
            logger.info("Created resource %s.", resource_path)  
        except ClientError:  
            logger.exception("Couldn't create resource %s.", resource_path)  
            raise  
        else:  
            return resource_id
```

- For API details, see [CreateResource](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an API Gateway REST API using an AWS SDK

The following code examples show how to delete an API Gateway REST API.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteAPI( ApiGatewayClient apiGateway, String restApiId) {  
  
    try {  
        DeleteRestApiRequest request = DeleteRestApiRequest.builder()  
            .restApiId(restApiId)  
            .build();  
  
        apiGateway.deleteRestApi(request);  
        System.out.println("The API was successfully deleted");  
  
    } catch (ApiGatewayException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteRestApi](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:  
    """  
    Encapsulates Amazon API Gateway functions that are used to create a REST API  
    that  
    integrates with another AWS service.  
    """  
    def __init__(self, apig_client):  
        """  
        :param apig_client: A Boto3 API Gateway client.  
        """  
        self.apig_client = apig_client  
        self.api_id = None  
        self.root_id = None  
        self.stage = None  
  
    def delete_rest_api(self):  
        """  
        Deletes a REST API, including all of its resources and configuration.  
        """  
        try:  
            self.apig_client.delete_rest_api(restApiId=self.api_id)  
            logger.info("Deleted REST API %s.", self.api_id)  
            self.api_id = None  
        except ClientError:  
            logger.exception("Couldn't delete REST API %s.", self.api_id)  
            raise
```

- For API details, see [DeleteRestApi](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an API Gateway deployment using an AWS SDK

The following code example shows how to delete a deployment.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteSpecificDeployment(ApiGatewayClient apiGateway, String
restApiId, String deploymentId) {

    try {
        DeleteDeploymentRequest request = DeleteDeploymentRequest.builder()
            .restApiId(restApiId)
            .deploymentId(deploymentId)
            .build();

        apiGateway.deleteDeployment(request);
        System.out.println("Deployment was deleted" );
    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteDeployment](#) in *AWS SDK for Java 2.x API Reference*.

## Deploy an API Gateway REST API using an AWS SDK

The following code examples show how to deploy an API Gateway REST API.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createNewDeployment(ApiGatewayClient apiGateway, String
restApiId, String stageName) {

    try {
        CreateDeploymentRequest request = CreateDeploymentRequest.builder()
            .restApiId(restApiId)
            .description("Created using the AWS API Gateway Java API")
            .stageName(stageName)
            .build();
    }
```

```
        CreateDeploymentResponse response =
    apiGateway.createDeployment(request);
    System.out.println("The id of the deployment is "+response.id());
    return response.id();

} catch (ApiGatewayException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "" ;
}
```

- For API details, see [CreateDeployment](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
    that
    integrates with another AWS service.
    """
    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None

    def deploy_api(self, stage_name):
        """
        Deploys a REST API. After a REST API is deployed, it can be called from any
        REST client, such as the Python Requests package or Postman.

        :param stage_name: The stage of the API to deploy, such as 'test'.
        :return: The base URL of the deployed REST API.
        """
        try:
            self.apig_client.create_deployment(
                restApiId=self.api_id, stageName=stage_name)
            self.stage = stage_name
            logger.info("Deployed stage %s.", stage_name)
        except ClientError:
            logger.exception("Couldn't deploy stage %s.", stage_name)
            raise
        else:
            return self.api_url()

    def api_url(self, resource=None):
        """
        Builds the REST API URL from its parts.

        :param resource: The resource path to append to the base URL.
        """


```

```
:return: The REST URL to the specified resource.  
"""\n    url = f'https://{{self.api_id}}.execute-api.\n{{self.apig_client.meta.region_name}}'\n        f'.amazonaws.com/{{self.stage}}')\n    if resource is not None:\n        url = f'{url}/{resource}'\n    return url
```

- For API details, see [CreateDeployment](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get API Gateway REST resources using an AWS SDK

The following code example shows how to get API Gateway REST resources.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:  
    """  
        Encapsulates Amazon API Gateway functions that are used to create a REST API  
        that  
        integrates with another AWS service.  
    """  
    def __init__(self, apig_client):  
        """  
            :param apig_client: A Boto3 API Gateway client.  
        """  
        self.apig_client = apig_client  
        self.api_id = None  
        self.root_id = None  
        self.stage = None  
  
    def create_rest_api(self, api_name):  
        """  
            Creates a REST API on API Gateway. The default API has only a root resource  
            and no HTTP methods.  
  
            :param api_name: The name of the API. This descriptive name is not used in  
                the API path.  
            :return: The ID of the newly created API.  
        """  
        try:  
            result = self.apig_client.create_rest_api(name=api_name)  
            self.api_id = result['id']  
            logger.info("Created REST API %s with ID %s.", api_name, self.api_id)  
        except ClientError:  
            logger.exception("Couldn't create REST API %s.", api_name)  
            raise  
  
        try:  
            result = self.apig_client.get_resources(restApiId=self.api_id)  
            self.root_id = next(  
                item for item in result['items'] if item['path'] == '/')['id']  
        except ClientError:  
            logger.exception("Couldn't get resources for API %s.", self.api_id)
```

```
        raise
    except StopIteration as err:
        logger.exception("No root resource found in API %s.", self.api_id)
        raise ValueError from err

    return self.api_id
```

- For API details, see [GetResources](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get the base path mapping for a custom domain name in API Gateway

The following code example shows how to get an API Gateway base path mapping.

PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;

/* /////////////////////////////////
 * Purpose: Gets the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 * API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 * URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 *
 * Returns: The base path mapping, if available; otherwise, the error message.
 * ////////////////////////////// */
function getBasePathMapping($apiGatewayClient, $basePath, $domainName)
{
    try {
        $result = $apiGatewayClient->getBasePathMapping([
            'basePath' => $basePath,
            'domainName' => $domainName,
        ]);
        return 'The base path mapping\'s effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function getsTheBasePathMapping()
{
```

```
$apiGatewayClient = new ApiGatewayClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2015-07-09'
]);

echo getBasePathMapping($apiGatewayClient, '(none)', 'example.com');

// Uncomment the following line to run this code in an AWS account.
// getsTheBasePathMapping();
```

- For API details, see [GetBasePathMapping](#) in *AWS SDK for PHP API Reference*.

## Integrate an API Gateway method with an AWS service using an AWS SDK

The following code example shows how to integrate an API Gateway method with an AWS service.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
    that
        integrates with another AWS service.
    """
    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None

    def add_integration_method(
            self, resource_id, rest_method, service_endpoint_prefix,
            service_action,
            service_method, role_arn, mapping_template):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated
        with
                                         this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                             'GetItem'.
        :param service_method: The HTTP method of the service request, such as
                             POST.
    
```

```
:param role_arn: The Amazon Resource Name (ARN) of a role that grants API
    Gateway permission to use the specified action with the
    service.
:param mapping_template: A mapping template that is used to translate REST
    elements, such as query parameters, to the request
    body format required by the service.
"""
service_uri = (f'arn:aws:apigateway:{self.apig_client.meta.region_name}'
               f'{service_endpoint_prefix}:action/{service_action}')
try:
    self.apig_client.put_method(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        authorizationType='NONE')
    self.apig_client.put_method_response(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        statusCode='200',
        responseModels={'application/json': 'Empty'})
    logger.info("Created %s method for resource %s.", rest_method,
    resource_id)
except ClientError:
    logger.exception(
        "Couldn't create %s method for resource %s.", rest_method,
    resource_id)
    raise

try:
    self.apig_client.put_integration(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        type='AWS',
        integrationHttpMethod=service_method,
        credentials=role_arn,
        requestTemplates={'application/json':
    json.dumps(mapping_template)},
        uri=service_uri,
        passthroughBehavior='WHEN_NO_TEMPLATES')
    self.apig_client.put_integration_response(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        statusCode='200',
        responseTemplates={'application/json': ''})
    logger.info(
        "Created integration for resource %s to service URI %s.",
    resource_id,
    service_uri)
except ClientError:
    logger.exception(
        "Couldn't create integration for resource %s to service URI %s.",
        resource_id, service_uri)
    raise
```

- For API details, see [PutIntegration](#) in *AWS SDK for Python (Boto3) API Reference*.

## List API Gateway REST APIs using an AWS SDK

The following code examples show how to list API Gateway REST APIs.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:  
    """  
        Encapsulates Amazon API Gateway functions that are used to create a REST API  
        that  
        integrates with another AWS service.  
    """  
    def __init__(self, apig_client):  
        """  
            :param apig_client: A Boto3 API Gateway client.  
        """  
        self.apig_client = apig_client  
        self.api_id = None  
        self.root_id = None  
        self.stage = None  
  
    def get_rest_api_id(self, api_name):  
        """  
            Gets the ID of a REST API from its name by searching the list of REST APIs  
            for the current account. Because names need not be unique, this returns  
            only  
            the first API with the specified name.  
  
            :param api_name: The name of the API to look up.  
            :return: The ID of the specified API.  
        """  
        try:  
            rest_api = None  
            paginator = self.apig_client.getPaginator('get_rest_apis')  
            for page in paginator.paginate():  
                rest_api = next(  
                    item for item in page['items'] if item['name'] == api_name),  
            None)  
            if rest_api is not None:  
                break  
            self.api_id = rest_api['id']  
            logger.info("Found ID %s for API %s.", rest_api['id'], api_name)  
        except ClientError:  
            logger.exception("Couldn't find ID for API %s.", api_name)  
            raise  
        else:  
            return rest_api['id']
```

- For API details, see [GetRestApis in AWS SDK for Python \(Boto3\) API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Displays the Amazon API Gateway REST APIs in the Region.

```
async fn show_apis(client: &Client) -> Result<(), Error> {
    let resp = client.get_rest_apis().send().await?;

    for api in resp.items().unwrap_or_default() {
        println!("ID:      {}", api.id().unwrap_or_default());
        println!("Name:    {}", api.name().unwrap_or_default());
        println!("Description: {}", api.description().unwrap_or_default());
        println!("Version:  {}", api.version().unwrap_or_default());
        println!(
            "Created:  {}",
            api.created_date().unwrap().to_chrono_utc()
        );
        println!();
    }
    Ok(())
}
```

- For API details, see [GetRestApis](#) in *AWS SDK for Rust API reference*.

## List the base path mapping for a custom domain name in API Gateway

The following code example shows how to list an API Gateway base path mapping.

PHP

### SDK for PHP

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;

/* /////////////////////////////////
 * Purpose: Lists the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $domainName: The custom domain name for the base path mappings.
 *
 * Returns: Information about the base path mappings, if available;
 * otherwise, the error message.
 */
```

```
* ///////////////////////////////*/
function listBasePathMappings($apiGatewayClient, $domainName)
{
    try {
        $result = $apiGatewayClient->getbasePathMappings([
            'domainName' => $domainName
        ]);
        return 'The base path mapping(s) effective URI is: ' .
            $result['@metadata']['effectiveUri'];
    } catch (AwsException $e) {
        return 'Error: ' . $e['message'];
    }
}

function listThebasePathMappings()
{
    $apiGatewayClient = new ApiGatewayClient([
        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2015-07-09'
    ]);

    echo listBasePathMappings($apiGatewayClient, 'example.com');
}

// Uncomment the following line to run this code in an AWS account.
// listThebasePathMappings();
```

- For API details, see [ListBasePathMappings](#) in *AWS SDK for PHP API Reference*.

## Update the base path mapping for a custom domain name in API Gateway

The following code example shows how to update an API Gateway base path mapping.

PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;

/* //////////////////////////////*/
/*
 * Purpose: Updates the base path mapping for a custom domain name
 * in Amazon API Gateway.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
```

```
* - $patchOperations: The base path update operations to apply.  
*  
* Returns: Information about the updated base path mapping, if available;  
* otherwise, the error message.  
* ///////////////////////////////// *//  
  
function updateBasePathMapping($apiGatewayClient, $basePath, $domainName,  
    $patchOperations)  
{  
    try {  
        $result = $apiGatewayClient->updateBasePathMapping([  
            'basePath' => $basePath,  
            'domainName' => $domainName,  
            'patchOperations' => $patchOperations  
        ]);  
        return 'The updated base path\'s URI is: ' .  
            $result['@metadata']['effectiveUri'];  
    } catch (AwsException $e) {  
        return 'Error: ' . $e['message'];  
    }  
}  
  
function updateTheBasePathMapping()  
{  
    $patchOperations = array([  
        'op' => 'replace',  
        'path' => '/stage',  
        'value' => 'stage2'  
    ]);  
  
    $apiGatewayClient = new ApiGatewayClient([  
        'profile' => 'default',  
        'region' => 'us-east-1',  
        'version' => '2015-07-09'  
    ]);  
  
    echo updateBasePathMapping(  
        $apiGatewayClient,  
        '(none)',  
        'example.com',  
        $patchOperations);  
}  
  
// Uncomment the following line to run this code in an AWS account.  
// updateTheBasePathMapping();
```

- For API details, see [UpdateBasePathMapping](#) in *AWS SDK for PHP API Reference*.

## Scenarios for API Gateway using AWS SDKs

The following code examples show how to use Amazon API Gateway with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Create and deploy a REST API using an AWS SDK \(p. 41\)](#)

## Create and deploy a REST API using an AWS SDK

The following code example shows how to:

- Create a REST API served by API Gateway.
- Add resources to the REST API to represent a user profile.
- Add integration methods so that the REST API uses a DynamoDB table to store user profile data.
- Send HTTP requests to the REST API to add and retrieve user profiles.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps API Gateway operations.

```
import argparse
import json
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)

class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API
    that
        integrates with another AWS service.
    """
    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None

    def create_rest_api(self, api_name):
        """
        Creates a REST API on API Gateway. The default API has only a root resource
        and no HTTP methods.

        :param api_name: The name of the API. This descriptive name is not used in
                         the API path.
        :return: The ID of the newly created API.
        """
        try:
            result = self.apig_client.create_rest_api(name=api_name)
            self.api_id = result['id']
            logger.info("Created REST API %s with ID %s.", api_name, self.api_id)
        except ClientError:
            logger.exception("Couldn't create REST API %s.", api_name)
            raise

        try:
            result = self.apig_client.get_resources(restApiId=self.api_id)
            self.root_id = next(
                item for item in result['items'] if item['path'] == '/')['id']
        except ClientError:
```

```
        logger.exception("Couldn't get resources for API %s.", self.api_id)
        raise
    except StopIteration as err:
        logger.exception("No root resource found in API %s.", self.api_id)
        raise ValueError from err

    return self.api_id

def add_rest_resource(self, parent_id, resource_path):
    """
    Adds a resource to a REST API.

    :param parent_id: The ID of the parent resource.
    :param resource_path: The path of the new resource, relative to the parent.
    :return: The ID of the new resource.
    """
    try:
        result = self.apig_client.create_resource(
            restApiId=self.api_id, parentId=parent_id, pathPart=resource_path)
        resource_id = result['id']
        logger.info("Created resource %s.", resource_path)
    except ClientError:
        logger.exception("Couldn't create resource %s.", resource_path)
        raise
    else:
        return resource_id

    def add_integration_method(
            self, resource_id, rest_method, service_endpoint_prefix,
            service_action,
            service_method, role_arn, mapping_template):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated
        with
                                         this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                               'GetItem'.
        :param service_method: The HTTP method of the service request, such as
                               POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                        Gateway permission to use the specified action with the
                        service.
        :param mapping_template: A mapping template that is used to translate REST
                               elements, such as query parameters, to the request
                               body format required by the service.
        """
        service_uri = (f'arn:aws:apigateway:{self.apig_client.meta.region_name}'
                      f':{service_endpoint_prefix}:action/{service_action}')
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType='NONE')
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode='200',
                responseModels={'application/json': 'Empty'})
        except ClientError:
            logger.exception("Couldn't add integration method %s.", service_uri)
            raise
    
```

```
        logger.info("Created %s method for resource %s.", rest_method,
resource_id)
    except ClientError:
        logger.exception(
            "Couldn't create %s method for resource %s.", rest_method,
resource_id)
        raise

    try:
        self.apig_client.put_integration(
            restApiId=self.api_id,
            resourceId=resource_id,
            httpMethod=rest_method,
            type='AWS',
            integrationHttpMethod=service_method,
            credentials=role_arn,
            requestTemplates={'application/json':
json.dumps(mapping_template)},
            uri=service_uri,
            passthroughBehavior='WHEN_NO_TEMPLATES')
        self.apig_client.put_integration_response(
            restApiId=self.api_id,
            resourceId=resource_id,
            httpMethod=rest_method,
            statusCode='200',
            responseTemplates={'application/json': ''})
        logger.info(
            "Created integration for resource %s to service URI %s.",
resource_id,
            service_uri)
    except ClientError:
        logger.exception(
            "Couldn't create integration for resource %s to service URI %s.",
            resource_id, service_uri)
        raise

def deploy_api(self, stage_name):
    """
    Deploys a REST API. After a REST API is deployed, it can be called from any
    REST client, such as the Python Requests package or Postman.

    :param stage_name: The stage of the API to deploy, such as 'test'.
    :return: The base URL of the deployed REST API.
    """
    try:
        self.apig_client.create_deployment(
            restApiId=self.api_id, stageName=stage_name)
        self.stage = stage_name
        logger.info("Deployed stage %s.", stage_name)
    except ClientError:
        logger.exception("Couldn't deploy stage %s.", stage_name)
        raise
    else:
        return self.api_url()

def api_url(self, resource=None):
    """
    Builds the REST API URL from its parts.

    :param resource: The resource path to append to the base URL.
    :return: The REST URL to the specified resource.
    """
    url = (f'https://{{self.api_id}}.execute-api.{{self.apig_client.meta.region_name}}'
           f'.amazonaws.com/{{self.stage}}')
```

```
if resource is not None:  
    url = f'{url}/{resource}'  
return url
```

Deploy a REST API and call it with the Requests package.

```
def usage_demo(table_name, role_name, rest_api_name):  
    """  
    Demonstrates how to used API Gateway to create and deploy a REST API, and how  
    to use the Requests package to call it.  
  
    :param table_name: The name of the demo DynamoDB table.  
    :param role_name: The name of the demo role that grants API Gateway permission  
    to  
        call DynamoDB.  
    :param rest_api_name: The name of the demo REST API created by the demo.  
    """  
    gateway = ApiGatewayToService(boto3.client('apigateway'))  
    role = boto3.resource('iam').Role(role_name)  
  
    print("Creating REST API in API Gateway.")  
    gateway.create_rest_api(rest_api_name)  
  
    print("Adding resources to the REST API.")  
    profiles_id = gateway.add_rest_resource(gateway.root_id, 'profiles')  
    username_id = gateway.add_rest_resource(profiles_id, '{username}')  
  
    # The DynamoDB service requires that all integration requests use POST.  
    print("Adding integration methods to read and write profiles in Amazon  
    DynamoDB.")  
    gateway.add_integration_method(  
        profiles_id, 'GET', 'dynamodb', 'Scan', 'POST', role.arn,  
        {'TableName': table_name})  
    gateway.add_integration_method(  
        profiles_id, 'POST', 'dynamodb', 'PutItem', 'POST', role.arn, {  
            "TableName": table_name,  
            "Item": {  
                "username": {"S": "$input.path('$.username')"},  
                "name": {"S": "$input.path('$.name')"},  
                "title": {"S": "$input.path('$.title')"}}})  
    gateway.add_integration_method(  
        username_id, 'GET', 'dynamodb', 'GetItem', 'POST', role.arn, {  
            "TableName": table_name,  
            "Key": {"username": {"S": "$method.request.path.username"}}})  
  
    stage = 'test'  
    print(f"Deploying the {stage} stage.")  
    gateway.deploy_api(stage)  
  
    profiles_url = gateway.api_url('profiles')  
    print(f"Using the Requests package to post some people to the profiles REST API  
    at "  
         f"[{profiles_url}].")  
    requests.post(profiles_url, json={  
        'username': 'will', 'name': 'William Shakespeare', 'title': 'playwright'})  
    requests.post(profiles_url, json={  
        'username': 'ludwig', 'name': 'Ludwig van Beethoven', 'title': 'composer'})  
    requests.post(profiles_url, json={  
        'username': 'jane', 'name': 'Jane Austen', 'title': 'author'})  
    print("Getting the list of profiles from the REST API.")  
    profiles = requests.get(profiles_url).json()  
    pprint(profiles)  
    print(f"Getting just the profile for username 'jane' (URL: {profiles_url}/  
    jane).")
```

```
jane = requests.get(f'{profiles_url}/jane').json()  
pprint(jane)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateDeployment](#)
  - [CreateResource](#)
  - [CreateRestApi](#)
  - [DeleteRestApi](#)
  - [GetResources](#)
  - [GetRestApis](#)
  - [PutIntegration](#)
  - [PutIntegrationResponse](#)
  - [PutMethod](#)
  - [PutMethodResponse](#)

## Cross-service examples for API Gateway using AWS SDKs

The following code examples show how to use Amazon API Gateway with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Create an API Gateway REST API to track COVID-19 data \(p. 46\)](#)
- [Create a lending library REST API \(p. 47\)](#)
- [Create a websocket chat application with API Gateway \(p. 47\)](#)
- [Use API Gateway to invoke a Lambda function \(p. 48\)](#)

### Create an API Gateway REST API to track COVID-19 data

The following code example shows how to create a REST API that simulates a system to track daily cases of COVID-19 in the United States, using fictional data.

#### Python

##### SDK for Python (Boto3)

Shows how to use AWS Chalice with the AWS SDK for Python (Boto3) to create a serverless REST API that uses Amazon API Gateway, AWS Lambda, and Amazon DynamoDB. The REST API simulates a system that tracks daily cases of COVID-19 in the United States, using fictional data. Learn how to:

- Use AWS Chalice to define routes in Lambda functions that are called to handle REST requests that come through API Gateway.
- Use Lambda functions to retrieve and store data in a DynamoDB table to serve REST requests.
- Define table structure and security role resources in an AWS CloudFormation template.
- Use AWS Chalice and CloudFormation to package and deploy all necessary resources.
- Use CloudFormation to clean up all created resources.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- API Gateway
- AWS CloudFormation
- DynamoDB
- Lambda

## Create a lending library REST API

The following code example shows how to create a lending library where patrons can borrow and return books by using a REST API backed by an Amazon Aurora database.

Python

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with the Amazon Relational Database Service (Amazon RDS) API and AWS Chalice to create a REST API backed by an Amazon Aurora database. The web service is fully serverless and represents a simple lending library where patrons can borrow and return books. Learn how to:

- Create and manage a serverless Aurora database cluster.
- Use AWS Secrets Manager to manage database credentials.
- Implement a data storage layer that uses Amazon RDS to move data into and out of the database.
- Use AWS Chalice to deploy a serverless REST API to Amazon API Gateway and AWS Lambda.
- Use the Requests package to send requests to the web service.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- API Gateway
- Lambda
- Amazon RDS
- Secrets Manager

## Create a websocket chat application with API Gateway

The following code example shows how to create a chat application that is served by a websocket API built on Amazon API Gateway.

Python

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with Amazon API Gateway V2 to create a websocket API that integrates with AWS Lambda and Amazon DynamoDB.

- Create a websocket API served by API Gateway.
- Define a Lambda handler that stores connections in DynamoDB and posts messages to other chat participants.
- Connect to the websocket chat application and send messages with the Websockets package.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- API Gateway
- DynamoDB
- Lambda

## Use API Gateway to invoke a Lambda function

The following code examples show how to create an AWS Lambda function invoked by Amazon API Gateway.

Java

### SDK for Java 2.x

Shows how to create an AWS Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create a Lambda function invoked by Amazon API Gateway that scans an Amazon DynamoDB table for work anniversaries and uses Amazon Simple Notification Service (Amazon SNS) to send a text message to your employees that congratulates them at their one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

### SDK for JavaScript V3

Shows how to create an AWS Lambda function by using the Lambda JavaScript runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create a Lambda function invoked by Amazon API Gateway that scans an Amazon DynamoDB table for work anniversaries and uses Amazon Simple Notification Service (Amazon SNS) to send a text message to your employees that congratulates them at their one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

### Services used in this example

- API Gateway
- DynamoDB
- Lambda

- Amazon SNS

Python

#### SDK for Python (Boto3)

This example shows how to create and use an Amazon API Gateway REST API that targets an AWS Lambda function. The Lambda handler demonstrates how to route based on HTTP methods; how to get data from the query string, header, and body; and how to return a JSON response.

- Deploy a Lambda function.
- Create an API Gateway REST API.
- Create a REST resource that targets the Lambda function.
- Grant permission to let API Gateway invoke the Lambda function.
- Use the Requests package to send requests to the REST API.
- Clean up all resources created during the demo.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- API Gateway
- Lambda

## Code examples for API Gateway Management API using AWS SDKs

The following code examples show how to use Amazon API Gateway Management API with an AWS software development kit (SDK).

#### Code examples

- [Actions for API Gateway Management API using AWS SDKs \(p. 49\)](#)
  - [Send the provided data to the specified connection using an AWS SDK \(p. 49\)](#)

## Actions for API Gateway Management API using AWS SDKs

The following code examples show how to use Amazon API Gateway Management API with AWS SDKs. Each example calls an individual service function.

#### Examples

- [Send the provided data to the specified connection using an AWS SDK \(p. 49\)](#)

## Send the provided data to the specified connection using an AWS SDK

The following code example shows how to send data to a connection.

Rust

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn send_data(
    client: &aws_sdk_apigatewaymanagement::Client,
    con_id: &str,
    data: &str,
) -> Result<(), aws_sdk_apigatewaymanagement::Error> {
    client
        .post_to_connection()
        .connection_id(con_id)
        .data(Blob::new(data))
        .send()
        .await?;

    Ok(())
}

let uri = format!(
    "https://{}.{execute-api.{region}.amazonaws.com/{stage}}",
    api_id = api_id,
    region = region,
    stage = stage
)
.parse()
.expect("could not construct valid URI for endpoint");
let endpoint = Endpoint::immutable(uri);

let shared_config =
aws_config::from_env().region(region_provider).load().await;
let api_management_config = config::Builder::from(&shared_config)
    .endpoint_resolver(endpoint)
    .build();
let client = Client::from_conf(api_management_config);
```

- For API details, see [PostToConnection](#) in *AWS SDK for Rust API reference*.

# Code examples for Application Auto Scaling using AWS SDKs

The following code examples show how to use Application Auto Scaling with an AWS software development kit (SDK).

## Code examples

- [Actions for Application Auto Scaling using AWS SDKs \(p. 51\)](#)
  - [Describe Application Auto Scaling scaling policies using an AWS SDK \(p. 51\)](#)

## Actions for Application Auto Scaling using AWS SDKs

The following code examples show how to use Application Auto Scaling with AWS SDKs. Each example calls an individual service function.

### Examples

- [Describe Application Auto Scaling scaling policies using an AWS SDK \(p. 51\)](#)

## Describe Application Auto Scaling scaling policies using an AWS SDK

The following code example shows how to describe Application Auto Scaling scaling policies for the specified service namespace.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_policies(client: &Client) -> Result<(), Error> {
    let response = client
        .describe_scaling_policies()
        .service_namespace(ServiceNamespace::Ec2)
        .send()
        .await?;
    if let Some(policies) = response.scaling_policies() {
        println!("Auto Scaling Policies:");
        for policy in policies {
            println!("{:?}", policy);
        }
    }
    println!("Next token: {:?}", response.next_token());
    Ok(())
}
```

- For API details, see [DescribeScalingPolicies](#) in *AWS SDK for Rust API reference*.

## Code examples for Application Recovery Controller using AWS SDKs

The following code examples show how to use Amazon Route 53 Application Recovery Controller with an AWS software development kit (SDK).

### Code examples

- [Actions for Application Recovery Controller using AWS SDKs \(p. 52\)](#)
  - [Get the state of an Application Recovery Controller routing control using an AWS SDK \(p. 52\)](#)
  - [Update the state of an Application Recovery Controller routing control using an AWS SDK \(p. 53\)](#)

## Actions for Application Recovery Controller using AWS SDKs

The following code examples show how to use Amazon Route 53 Application Recovery Controller with AWS SDKs. Each example calls an individual service function.

### Examples

- [Get the state of an Application Recovery Controller routing control using an AWS SDK \(p. 52\)](#)
- [Update the state of an Application Recovery Controller routing control using an AWS SDK \(p. 53\)](#)

## Get the state of an Application Recovery Controller routing control using an AWS SDK

The following code examples show how to get the state of an Application Recovery Controller routing control.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static GetRoutingControlStateResponse
getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
                      String
routingControlArn) {
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region())).build();
            return client.getRoutingControlState(
                GetRoutingControlStateRequest.builder()
                    .routingControlArn(routingControlArn).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}
```

- For API details, see [GetRoutingControlState](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import boto3

def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the
    specified
    cluster endpoint URL and AWS Region.

    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
        'route53-recovery-cluster',
        endpoint_url=cluster_endpoint['Endpoint'],
        region_name=cluster_endpoint['Region'])

def get_routing_control_state(routing_control_arn, cluster_endpoints):
    """
    Gets the state of a routing control. Cluster endpoints are tried in
    sequence until the first successful response is received.

    :param routing_control_arn: The ARN of the routing control to look up.
    :param cluster_endpoints: The list of cluster endpoints to query.
    :return: The routing control state response.
    """
    for cluster_endpoint in cluster_endpoints:
        try:
            recovery_client = create_recovery_client(cluster_endpoint)
            response = recovery_client.get_routing_control_state(
                RoutingControlArn=routing_control_arn)
            return response
        except Exception as error:
            print(error)
```

- For API details, see [GetRoutingControlState](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update the state of an Application Recovery Controller routing control using an AWS SDK

The following code examples show how to update the state of an Application Recovery Controller routing control.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static UpdateRoutingControlStateResponse
updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
String routingControlArn,
String routingControlState) {
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region()))
                .build();
            return client.updateRoutingControlState(
                UpdateRoutingControlStateRequest.builder()
                    .routingControlArn(routingControlArn).routingControlState(routingControlState).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}
```

- For API details, see [UpdateRoutingControlState](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import boto3

def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the
    specified
    cluster endpoint URL and AWS Region.

    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
        'route53-recovery-cluster',
        endpoint_url=cluster_endpoint['Endpoint'],
        region_name=cluster_endpoint['Region'])

def update_routing_control_state(
    routing_control_arn, cluster_endpoints, routing_control_state):
    """
    Updates the state of a routing control. Cluster endpoints are tried in
    sequence until the first successful response is received.

    :param routing_control_arn: The ARN of the routing control to update the state
    for.
    :param cluster_endpoints: The list of cluster endpoints to try.
```

```
:param routing_control_state: The new routing control state.  
:return: The routing control update response.  
"""  
for cluster_endpoint in cluster_endpoints:  
    try:  
        recovery_client = create_recovery_client(cluster_endpoint)  
        response = recovery_client.update_routing_control_state(  
            RoutingControlArn=routing_control_arn,  
            RoutingControlState=routing_control_state)  
        return response  
    except Exception as error:  
        print(error)
```

- For API details, see [UpdateRoutingControlState](#) in *AWS SDK for Python (Boto3) API Reference*.

## Code examples for Audit Manager using AWS SDKs

The following code examples show how to use AWS Audit Manager with an AWS software development kit (SDK).

### Code examples

- [Scenarios for Audit Manager using AWS SDKs \(p. 55\)](#)
  - [Create an Audit Manager custom framework from an AWS Config conformance pack using an AWS SDK \(p. 55\)](#)
  - [Create an Audit Manager custom framework that contains Security Hub controls using an AWS SDK \(p. 58\)](#)
  - [Create an Audit Manager assessment report that contains one day of evidence using an AWS SDK \(p. 60\)](#)

## Scenarios for Audit Manager using AWS SDKs

The following code examples show how to use AWS Audit Manager with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Create an Audit Manager custom framework from an AWS Config conformance pack using an AWS SDK \(p. 55\)](#)
- [Create an Audit Manager custom framework that contains Security Hub controls using an AWS SDK \(p. 58\)](#)
- [Create an Audit Manager assessment report that contains one day of evidence using an AWS SDK \(p. 60\)](#)

## Create an Audit Manager custom framework from an AWS Config conformance pack using an AWS SDK

The following code example shows how to:

- Get a list of AWS Config conformance packs.
- Create an Audit Manager custom control for each managed rule in a conformance pack.
- Create an Audit Manager custom framework that contains the controls.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class ConformancePack:
    def __init__(self, config_client, auditmanager_client):
        self.config_client = config_client
        self.auditmanager_client = auditmanager_client

    def get_conformance_pack(self):
        """
        Return a selected conformance pack from the list of conformance packs.

        :return: selected conformance pack
        """
        try:
            conformance_packs = self.config_client.describe_conformance_packs()
            print("Number of conformance packs fetched: ",
                  len(conformance_packs.get("ConformancePackDetails")))
            print("Fetched the following conformance packs: ")
            all_cpack_names = [
                cp['ConformancePackName']
                for cp in conformance_packs.get("ConformancePackDetails")]
            for pack in all_cpack_names:
                print(f"\t{pack}")
            cpack_name = input(
                'Provide ConformancePackName that you want to create a custom '
                'framework for: ')
            if cpack_name not in all_cpack_names:
                print(f'{cpack_name} is not in the list of conformance packs!')
                print('Provide a conformance pack name from the available list of '
                      'conformance packs.')
                raise Exception("Invalid conformance pack")
            print('-' * 88)
        except ClientError:
            logger.exception("Couldn't select conformance pack.")
            raise
        else:
            return cpack_name

    def create_custom_controls(self, cpack_name):
        """
        Create custom controls for all managed AWS Config rules in a conformance
        pack.

        :param cpack_name: The name of the conformance pack to create controls for.
        :return: The list of custom control IDs.
        """
        try:
            rules_in_pack =
                self.config_client.describe_conformance_pack_compliance(
                    ConformancePackName=cpack_name)
            print('Number of rules in the conformance pack: ',
```

```

        len(rules_in_pack.get('ConformancePackRuleComplianceList'))))
for rule in rules_in_pack.get('ConformancePackRuleComplianceList'):
    print(f"\t{rule.get('ConfigRuleName')}")
print('-' * 88)
print('Creating a custom control for each rule and a custom framework '
      'consisting of these rules in Audit Manager.')
am_controls = []
for rule in rules_in_pack.get('ConformancePackRuleComplianceList'):
    config_rule = self.config_client.describe_config_rules(
        ConfigRuleNames=[rule.get('ConfigRuleName')])
    source_id = config_rule.get('ConfigRules')[0].get('Source',
{}).get(
    'SourceIdentifier')
    custom_control = self.auditmanager_client.create_control(
        name="Config-" + rule.get('ConfigRuleName'),
        controlMappingSources=[{
            'sourceName': 'ConfigRule',
            'sourceSetUpOption': 'System_Controls_Mapping',
            'sourceType': 'AWS_Config',
            'sourceKeyword': {
                'keywordInputType': 'SELECT_FROM_LIST',
                'keywordValue': source_id}]).get('control', {})
    am_controls.append({'id': custom_control.get('id')})
    print('Successfully created a control for each config rule.')
    print('-' * 88)
except ClientError:
    logger.exception("Failed to create custom controls.")
    raise
else:
    return am_controls

def create_custom_framework(self, cpack_name, am_control_ids):
    """
    Create a custom Audit Manager framework from a selected AWS Config
    conformance
    pack.

    :param cpack_name: The name of the conformance pack to create a framework
    from.
    :param am_control_ids: The IDs of the custom controls created from the
                           conformance pack.
    """
    try:
        print('Creating custom framework...')
        custom_framework =
self.auditmanager_client.create_assessment_framework(
            name='Config-Conformance-pack-' + cpack_name,
            controlSets=[{'name': cpack_name, 'controls': am_control_ids}])
        print(f"Successfully created the custom framework: ",
              f"[custom_framework.get('framework').get('name')]: ",
              f"[custom_framework.get('framework').get('id')]")
        print('-' * 88)
    except ClientError:
        logger.exception("Failed to create custom framework.")
        raise

def run_demo():
    print('-' * 88)
    print("Welcome to the AWS Audit Manager custom framework demo!")
    print('-' * 88)
    print("You can use this sample to select a conformance pack from AWS Config and"
          "
          "use AWS Audit Manager to create a custom control for all the managed "
          "rules under the conformance pack. A custom framework is also created "
          "with these controls.")

```

```
print('-' * 88)
conf_pack = ConformancePack(boto3.client('config'),
boto3.client('auditmanager'))
cpack_name = conf_pack.get_conformance_pack()
am_controls = conf_pack.create_custom_controls(cpack_name)
conf_pack.create_custom_framework(cpack_name, am_controls)

if __name__ == '__main__':
    run_demo()
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateAssessmentFramework](#)
  - [CreateControl](#)

## Create an Audit Manager custom framework that contains Security Hub controls using an AWS SDK

The following code example shows how to:

- Get a list of all standard controls that have Security Hub as their data source.
- Create an Audit Manager custom framework that contains the controls.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class SecurityHub:
    def __init__(self, auditmanager_client):
        self.auditmanager_client = auditmanager_client

    def get_sechub_controls(self):
        """
        Gets the list of controls that use Security Hub as their data source.

        :return: The list of Security Hub controls.
        """
        print('*'*88)
        next_token = None
        page = 1
        sechub_control_list = []
        while True:
            print("Page [" + str(page) + "]")
            if next_token is None:
                control_list = self.auditmanager_client.list_controls(
                    controlType='Standard',
```

```

        maxResults=100)
    else:
        control_list = self.auditmanager_client.list_controls(
            controlType='Standard',
            nextToken=next_token,
            maxResults=100)
        print('Total controls found:', len(control_list.get('controlMetadataList')))
        for control in control_list.get('controlMetadataList'):
            control_details = self.auditmanager_client.get_control(
                controlId=control.get('id')).get('control', {})
            if "AWS Security Hub" in control_details.get('controlSources'):
                sechub_control_list.append({'id': control_details.get('id')})
        next_token = control_list.get('nextToken')
        if not next_token:
            break
        page += 1
    print('Number of Security Hub controls found: ', len(sechub_control_list))
    return sechub_control_list

def create_custom_framework(self, am_controls):
    """
    Create a custom framework with a list of controls.

    :param am_controls: The list of controls to include in the framework.
    """
    try:
        print('Creating custom framework...')
        custom_framework =
self.auditmanager_client.create_assessment_framework(
            name='All Security Hub Controls Framework',
            controlSets=[{'name': "Security-Hub", 'controls': am_controls}])
        print(f"Successfully created the custom framework: "
              f"{custom_framework.get('framework').get('name')}: "
              f"{custom_framework.get('framework').get('id')}")
        print('-' * 88)
    except ClientError:
        logger.exception("Failed to create custom framework.")
        raise

def run_demo():
    print('-' * 88)
    print("Welcome to the AWS Audit Manager Security Hub demo!")
    print('-' * 88)
    print(" This script creates a custom framework with all Security Hub
controls.")
    print('-' * 88)
    sechub = SecurityHub(boto3.client('auditmanager'))
    am_controls = sechub.get_sechub_controls()
    sechub.create_custom_framework(am_controls)

if __name__ == '__main__':
    run_demo()

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateAssessmentFramework](#)
  - [GetControl](#)
  - [ListControls](#)

## Create an Audit Manager assessment report that contains one day of evidence using an AWS SDK

The following code example shows how to create an Audit Manager assessment report that contains one day of evidence.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import dateutil.parser
import logging
import time
import urllib.request
import uuid
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class AuditReport:
    def __init__(self, auditmanager_client):
        self.auditmanager_client = auditmanager_client

    def get_input(self):
        print('-' * 40)
        try:
            assessment_id = input('Provide assessment id [uuid]: ').lower()
            try:
                assessment_uuid = uuid.UUID(assessment_id)
            except ValueError:
                logger.error("Assessment Id is not a valid UUID: %s",
                            assessment_id)
                raise
            evidence_folder = input('Provide evidence date [yyyy-mm-dd]: ')
            try:
                evidence_date = dateutil.parser.parse(evidence_folder).date()
            except ValueError:
                logger.error("Invalid date : %s", evidence_folder)
                raise
            try:
                self.auditmanager_client.get_assessment(assessmentId=str(assessment_uuid))
            except ClientError:
                logger.exception("Couldn't get assessment %s.", assessment_uuid)
                raise
            except (ValueError, ClientError):
                return None, None
            else:
                return assessment_uuid, evidence_date
        def clear_staging(self, assessment_uuid, evidence_date):
            """
            Find all the evidence in the report and clear it.
            """
            next_token = None
```

```

page = 1
interested_folder_id_list = []
while True:
    print(f"Page [{page}]")
    if next_token is None:
        folder_list =
self.auditmanager_client.get_evidence_folders_by_assessment(
            assessmentId=str(assessment_uuid),
            maxResults=1000)
    else:
        folder_list =
self.auditmanager_client.get_evidence_folders_by_assessment(
            assessmentId=str(assessment_uuid),
            nextToken=next_token,
            maxResults=1000)
    folders = folder_list.get('evidenceFolders')
    print(f"Got {len(folders)} folders.")
    for folder in folders:
        folder_id = folder.get('id')
        if folder.get('name') == str(evidence_date):
            interested_folder_id_list.append(folder_id)
        if folder.get('assessmentReportSelectionCount') ==
folder.get('totalEvidence'):
            print(
                f"Removing folder from report selection :
{folder.get('name')} "
                f"{folder_id} {folder.get('controlId')}")
    self.auditmanager_client.disassociate_assessment_report_evidence(
        assessmentId=str(assessment_uuid),
        evidenceFolderId=folder_id)
    elif folder.get('assessmentReportSelectionCount') > 0:
        # Get all evidence in the folder and
        # add selected evidence in the selected_evidence_list.
        evidence_list =
self.auditmanager_client.get_evidence_by_evidence_folder(
            assessmentId=str(assessment_uuid),
            controlSetId=folder_id,
            evidenceFolderId=folder_id,
            maxResults=1000)
        selected_evidence_list = []
        for evidence in evidence_list.get('evidence'):
            if evidence.get('assessmentReportSelection') == 'Yes':
                selected_evidence_list.append(evidence.get('id'))
        print(f"Removing evidence report selection :
{folder.get('name')} "
              f"{len(selected_evidence_list)})")
    self.auditmanager_client.batch_disassociate_assessment_report_evidence(
        assessmentId=str(assessment_uuid),
        evidenceFolderId=folder_id,
        evidenceIds=selected_evidence_list)
    next_token = folder_list.get('nextToken')
    if not next_token:
        break
    page += 1
return interested_folder_id_list

def add_folder_to_staging(self, assessment_uuid, folder_id_list):
    print(f"Adding folders to report : {folder_id_list}")
    for folder in folder_id_list:
        self.auditmanager_client.associate_assessment_report_evidence_folder(
            assessmentId=str(assessment_uuid),
            evidenceFolderId=folder)

def get_report(self, assessment_uuid):

```

```

        report = self.auditmanager_client.create_assessment_report(
            name='ReportViaScript',
            description='testing',
            assessmentId=str(assessment_uuid))
        if self._is_report_generated(report.get('assessmentReport').get('id')):
            report_url = self.auditmanager_client.get_assessment_report_url(
                assessmentReportId=report.get('assessmentReport').get('id'),
                assessmentId=str(assessment_uuid))
            print(report_url.get('preSignedUrl'))
            urllib.request.urlretrieve(
                report_url.get('preSignedUrl').get('link'),
                report_url.get('preSignedUrl').get('hyperlinkName'))
            print(f"Report saved as
{report_url.get('preSignedUrl').get('hyperlinkName')}.")

        else:
            print("Report generation did not finish in 15 minutes.")
            print("Failed to download report. Go to the console and manually
download "
                  "the report.")

    def _is_report_generated(self, assessment_report_id):
        max_wait_time = 0
        while max_wait_time < 900:
            print(f"Checking status of the report {assessment_report_id}")
            report_list =
self.auditmanager_client.list_assessment_reports(maxResults=1)
            if (report_list.get('assessmentReports')[0].get('id') ==
assessment_report_id
                and report_list.get('assessmentReports')[0].get('status') ==
'COMPLETE'):
                return True
            print('Sleeping for 5 seconds...')
            time.sleep(5)
            max_wait_time += 5

    def run_demo():
        print('-' * 88)
        print("Welcome to the AWS Audit Manager samples demo!")
        print('-' * 88)
        print("This script creates an assessment report for an assessment with all the
"
              "evidence collected on the provided date.")
        print('-' * 88)

        report = AuditReport(boto3.client('auditmanager'))
        assessment_uuid, evidence_date = report.get_input()
        if assessment_uuid is not None and evidence_date is not None:
            folder_id_list = report.clear_staging(assessment_uuid, evidence_date)
            report.add_folder_to_staging(assessment_uuid, folder_id_list)
            report.get_report(assessment_uuid)

    if __name__ == '__main__':
        run_demo()

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [AssociateAssessmentReportEvidenceFolder](#)
  - [BatchDisassociateAssessmentReportEvidence](#)
  - [CreateAssessmentReport](#)
  - [DisassociateAssessmentReportEvidenceFolder](#)
  - [GetAssessment](#)

- [GetAssessmentReportUrl](#)
- [GetEvidenceByEvidenceFolder](#)
- [GetEvidenceFoldersByAssessment](#)
- [ListAssessmentReports](#)

## Code examples for Aurora using AWS SDKs

The following code examples show how to use Amazon Aurora with an AWS software development kit (SDK).

### Code examples

- [Actions for Aurora using AWS SDKs \(p. 63\)](#)
  - Create an Aurora DB cluster using an AWS SDK (p. 64)
  - Create an Aurora DB cluster parameter group using an AWS SDK (p. 66)
  - Create an Aurora DB cluster snapshot using an AWS SDK (p. 68)
  - Create a DB instance in an Aurora DB cluster using an AWS SDK (p. 70)
  - Delete an Aurora DB cluster using an AWS SDK (p. 72)
  - Delete an Aurora DB cluster parameter group using an AWS SDK (p. 74)
  - Delete an Aurora DB instance using an AWS SDK (p. 77)
  - Describe Aurora DB cluster parameter groups using an AWS SDK (p. 78)
  - Describe Aurora DB cluster snapshots using an AWS SDK (p. 80)
  - Describe Aurora DB clusters using an AWS SDK (p. 83)
  - Describe Aurora DB instances using an AWS SDK (p. 85)
  - Describe Aurora database engine versions using an AWS SDK (p. 88)
  - Describe options for Aurora DB instances using an AWS SDK (p. 90)
  - Describe parameters from an Aurora DB cluster parameter group using an AWS SDK (p. 92)
  - Update parameters in an Aurora DB cluster parameter group using an AWS SDK (p. 94)
- [Scenarios for Aurora using AWS SDKs \(p. 97\)](#)
  - Get started with Aurora DB clusters using an AWS SDK (p. 97)
- [Cross-service examples for Aurora using AWS SDKs \(p. 126\)](#)
  - Create an Aurora Serverless work item tracker (p. 126)

## Actions for Aurora using AWS SDKs

The following code examples show how to use Amazon Aurora with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create an Aurora DB cluster using an AWS SDK \(p. 64\)](#)
- [Create an Aurora DB cluster parameter group using an AWS SDK \(p. 66\)](#)
- [Create an Aurora DB cluster snapshot using an AWS SDK \(p. 68\)](#)
- [Create a DB instance in an Aurora DB cluster using an AWS SDK \(p. 70\)](#)
- [Delete an Aurora DB cluster using an AWS SDK \(p. 72\)](#)
- [Delete an Aurora DB cluster parameter group using an AWS SDK \(p. 74\)](#)
- [Delete an Aurora DB instance using an AWS SDK \(p. 77\)](#)

- [Describe Aurora DB cluster parameter groups using an AWS SDK \(p. 78\)](#)
- [Describe Aurora DB cluster snapshots using an AWS SDK \(p. 80\)](#)
- [Describe Aurora DB clusters using an AWS SDK \(p. 83\)](#)
- [Describe Aurora DB instances using an AWS SDK \(p. 85\)](#)
- [Describe Aurora database engine versions using an AWS SDK \(p. 88\)](#)
- [Describe options for Aurora DB instances using an AWS SDK \(p. 90\)](#)
- [Describe parameters from an Aurora DB cluster parameter group using an AWS SDK \(p. 92\)](#)
- [Update parameters in an Aurora DB cluster parameter group using an AWS SDK \(p. 94\)](#)

## Create an Aurora DB cluster using an AWS SDK

The following code examples show how to create an Aurora DB cluster.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createDBCluster(RdsClient rdsClient, String dbParameterGroupFamily, String dbName, String dbClusterIdentifier, String userName, String password) {
    try {
        CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
        .databaseName(dbName)
        .dbClusterIdentifier(dbClusterIdentifier)
        .dbClusterParameterGroupName(dbParameterGroupFamily)
        .engine("aurora-mysql")
        .masterUsername(userName)
        .masterUserPassword(password)
        .build();

        CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
        return response.dbCluster().dbClusterArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateDBCluster](#) in *AWS SDK for Java 2.x API Reference*.

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createDBCluster(dbParameterGroupFamilyVal: String?, dbName: String?,  
    dbClusterIdentifierVal: String?, userName: String?, password: String?): String? {  
    val clusterRequest = CreateDbClusterRequest {  
        databaseName = dbName  
        dbClusterIdentifier = dbClusterIdentifierVal  
        dbClusterParameterGroupName = dbParameterGroupFamilyVal  
        engine = "aurora-mysql"  
        masterUsername = userName  
        masterUserPassword = password  
    }  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val response = rdsClient.createDbCluster(clusterRequest)  
        return response.dbCluster?.dbClusterArn  
    }  
}
```

- For API details, see [CreateDBCluster](#) in *AWS SDK for Kotlin API reference*.

Python

**SDK for Python (Boto3)**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:  
    """Encapsulates Aurora DB cluster actions."""  
    def __init__(self, rds_client):  
        """  
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)  
        client.  
        """  
        self.rds_client = rds_client  
  
    @classmethod  
    def from_client(cls):  
        """  
        Instantiates this class from a Boto3 client.  
        """  
        rds_client = boto3.client('rds')  
        return cls(rds_client)  
  
    def create_db_cluster(  
        self, cluster_name, parameter_group_name, db_name, db_engine,  
        db_engine_version, admin_name, admin_password):  
        """  
        Creates a DB cluster that is configured to use the specified parameter  
        group.  
        The newly created DB cluster contains a database that uses the specified  
        engine and  
        engine version.  
        """  
        :param cluster_name: The name of the DB cluster to create.
```

```
        :param parameter_group_name: The name of the parameter group to associate
with
                           the DB cluster.
        :param db_name: The name of the database to create.
        :param db_engine: The database engine of the database that is created, such
as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
"""
try:
    response = self.rds_client.create_db_cluster(
        DatabaseName=db_name,
        DBClusterIdentifier=cluster_name,
        DBClusterParameterGroupName=parameter_group_name,
        Engine=db_engine,
        EngineVersion=db_engine_version,
        MasterUsername=admin_name,
        MasterUserPassword=admin_password)
    cluster = response['DBCluster']
except ClientError as err:
    logger.error(
        "Couldn't create database %s. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return cluster
```

- For API details, see [CreateDBCluster](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an Aurora DB cluster parameter group using an AWS SDK

The following code examples show how to create an Aurora DB cluster parameter group.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
        .dbClusterParameterGroupName(dbClusterGroupName)
        .dbParameterGroupFamily(dbParameterGroupFamily)
        .description("Created by using the AWS SDK for Java")
        .build();

        CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
        System.out.println("The group name is "+
response.dbClusterParameterGroup().dbClusterParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- For API details, see [CreateDBClusterParameterGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createDBClusterParameterGroup(dbClusterGroupNameVal: String?,
                                         dbParameterGroupFamilyVal: String?) {
    val groupRequest = CreateDbClusterParameterGroupRequest {
        dbClusterParameterGroupName = dbClusterGroupNameVal
        dbParameterGroupFamily = dbParameterGroupFamilyVal
        description = "Created by using the AWS SDK for Kotlin"
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- For API details, see [CreateDBClusterParameterGroup](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """

```

```
rds_client = boto3.client('rds')
return cls(rds_client)

def create_parameter_group(self, parameter_group_name, parameter_group_family,
                           description):
    """
        Creates a DB cluster parameter group that is based on the specified
        parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter group.
        :param parameter_group_family: The family that is used as the basis of the
                                       new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
    """
    try:
        response = self.rds_client.create_db_cluster_parameter_group(
            DBClusterParameterGroupName=parameter_group_name,
            DBParameterGroupFamily=parameter_group_family,
            Description=description)
    except ClientError as err:
        logger.error(
            "Couldn't create parameter group %s. Here's why: %s: %s",
            parameter_group_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response
```

- For API details, see [CreateDBClusterParameterGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an Aurora DB cluster snapshot using an AWS SDK

The following code examples show how to create an Aurora DB cluster snapshot.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
        .build();

        CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
        System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see [CreateDBClusterSnapshot](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createDBClusterSnapshot(dbInstanceClusterIdentifier: String?,
                                    dbSnapshotIdentifier: String?) {
    val snapshotRequest = CreateDbClusterSnapshotRequest {
        dbClusterIdentifier = dbInstanceClusterIdentifier
        dbClusterSnapshotIdentifier = dbSnapshotIdentifier
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- For API details, see [CreateDBClusterSnapshot](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        
```

```
"""
rds_client = boto3.client('rds')
return cls(rds_client)

def create_cluster_snapshot(self, snapshot_id, cluster_id):
    """
    Creates a snapshot of a DB cluster.

    :param snapshot_id: The ID to give the created snapshot.
    :param cluster_id: The DB cluster to snapshot.
    :return: Data about the newly created snapshot.
    """
    try:
        response = self.rds_client.create_db_cluster_snapshot(
            DBClusterSnapshotIdentifier=snapshot_id,
            DBClusterIdentifier=cluster_id)
        snapshot = response['DBClusterSnapshot']
    except ClientError as err:
        logger.error(
            "Couldn't create snapshot of %s. Here's why: %s: %s",
            cluster_id,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return snapshot
```

- For API details, see [CreateDBClusterSnapshot](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a DB instance in an Aurora DB cluster using an AWS SDK

The following code examples show how to create a DB instance in an Aurora DB cluster.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createDBInstanceCluster(RdsClient rdsClient,
                                             String dbInstanceIdentifier,
                                             String
dbInstanceClusterIdentifier,
                                             String instanceClass){

    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .engine("aurora-mysql")
        .dbInstanceClass(instanceClass)
        .build() ;

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceState());
        return response.dbInstance().dbInstanceArn();
```

```
        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createDBInstanceCluster(dbIdentifierVal: String?,
                                    dbClusterIdentifierVal: String?, instanceClassVal: String?): String? {
    val instanceRequest = CreateDbInstanceRequest {
        dbIdentifier = dbIdentifierVal
        dbClusterIdentifier = dbClusterIdentifierVal
        engine = "aurora-mysql"
        dbInstanceClass = instanceClassVal
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceState}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
```

```
def from_client(cls):
    """
    Instantiates this class from a Boto3 client.
    """
    rds_client = boto3.client('rds')
    return cls(rds_client)

    def create_instance_in_cluster(self, instance_id, cluster_id, db_engine,
instance_class):
        """
        Creates a database instance in an existing DB cluster. The first database
that is
            created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
                This must be compatible with the configured parameter
group
                    of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
    try:
        response = self.rds_client.create_db_instance(
            DBInstanceIdentifier=instance_id, DBClusterIdentifier=cluster_id,
            Engine=db_engine, DBInstanceClass=instance_class)
        db_inst = response['DBInstance']
    except ClientError as err:
        logger.error(
            "Couldn't create DB instance %s. Here's why: %s: %s",
            instance_id,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return db_inst
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an Aurora DB cluster using an AWS SDK

The following code examples show how to delete an Aurora DB cluster.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
    try {
        DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .skipFinalSnapshot(true)
```

```
        .build();

        rdsClient.deleteDBCluster(deleteDbClusterRequest);
        System.out.println(dbInstanceClusterIdentifier +" was deleted!");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteDBCluster](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest = DeleteDbClusterRequest {
        dbClusterIdentifier = dbInstanceClusterIdentifier
        skipFinalSnapshot = true
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

- For API details, see [DeleteDBCluster](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
```

```
def from_client(cls):
    """
    Instantiates this class from a Boto3 client.
    """
    rds_client = boto3.client('rds')
    return cls(rds_client)

def delete_db_cluster(self, cluster_name):
    """
    Deletes a DB cluster.

    :param cluster_name: The name of the DB cluster to delete.
    """
    try:
        self.rds_client.delete_db_cluster(
            DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True)
        logger.info("Deleted DB cluster %s.", cluster_name)
    except ClientError:
        logger.exception("Couldn't delete DB cluster %s.", cluster_name)
        raise
```

- For API details, see [DeleteDBCluster](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an Aurora DB cluster parameter group using an AWS SDK

The following code examples show how to delete an Aurora DB cluster parameter group.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deletedBClusterGroup( RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN) throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN ;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            isDataDel = false ;
            didFind = false;
            int index = 1;
            for (DBInstance instance: instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(clusterDBARN) == 0) {
                    System.out.println(clusterDBARN + " still exists");
                    didFind = true ;
                }
                if ((index == listSize) && (!didFind)) {
                    // Went through the entire list and did not find the
database ARN.
                    isDataDel = true;
                }
            }
        }
    }
}
```

```
        }
        Thread.sleep(sleepTime * 1000);
        index++;
    }
}

DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest.builder()
    .dbClusterParameterGroupName(dbClusterGroupName)
    .build();

rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
System.out.println(dbClusterGroupName +" was deleted.");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- For API details, see [DeleteDBClusterParameterGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
@Throws(InterruptedIOException::class)
suspend fun deleteDBClusterGroup(dbClusterGroupName: String, clusterDBARN: String)
{
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
                        database ARN.
                        isDataDel = true
                    }
                }
            }
        }
    }
}
```

```
        }
        delay(slTime * 1000)
        index++
    }
}
val clusterParameterGroupRequest = DeleteDbClusterParameterGroupRequest {
    dbClusterParameterGroupName = dbClusterGroupName
}

rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
println("$dbClusterGroupName was deleted.")
}
}
```

- For API details, see [DeleteDBClusterParameterGroup](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

- For API details, see [DeleteDBClusterParameterGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an Aurora DB instance using an AWS SDK

The following code examples show how to delete an Aurora DB instance.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteDatabaseInstance( RdsClient rdsClient, String dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .deleteAutomatedBackups(true)
        .skipFinalSnapshot(true)
        .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.println("The status of the database is " +
response.dbInstance().dbInstanceState());
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest = DeleteDbInstanceRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
        deleteAutomatedBackups = true
        skipFinalSnapshot = true
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
```

```
    val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
    print("The status of the database is
${response.dbInstance?.dbInstanceState}")
}
}
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id, SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True)
            db_inst = response['DBInstance']
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id, err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return db_inst
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe Aurora DB cluster parameter groups using an AWS SDK

The following code examples show how to describe Aurora DB cluster parameter groups.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
        DescribeDbClusterParameterGroupsRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .maxRecords(20)
            .build();

        List<DBClusterParameterGroup> groups =
        rdsClient.describeDBClusterParameterGroups(groupsRequest).dbClusterParameterGroups();
        for (DBClusterParameterGroup group: groups) {
            System.out.println("The group name is
"+group.dbClusterParameterGroupName());
            System.out.println("The group ARN is
"+group.dbClusterParameterGroupArn());
        }
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBClusterParameterGroups](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest = DescribeDbClusterParameterGroupsRequest {
        dbClusterParameterGroupName = dbClusterGroupName
        maxRecords = 20
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

- For API details, see [DescribeDBClusterParameterGroups](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:  
    """Encapsulates Aurora DB cluster actions."""  
    def __init__(self, rds_client):  
        """  
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)  
        client.  
        """  
        self.rds_client = rds_client  
  
    @classmethod  
    def from_client(cls):  
        """  
        Instantiates this class from a Boto3 client.  
        """  
        rds_client = boto3.client('rds')  
        return cls(rds_client)  
  
    def get_parameter_group(self, parameter_group_name):  
        """  
        Gets a DB cluster parameter group.  
  
        :param parameter_group_name: The name of the parameter group to retrieve.  
        :return: The requested parameter group.  
        """  
        try:  
            response = self.rds_client.describe_db_cluster_parameter_groups(  
                DBClusterParameterGroupName=parameter_group_name)  
            parameter_group = response['DBClusterParameterGroups'][0]  
        except ClientError as err:  
            if err.response['Error']['Code'] == 'DBParameterGroupNotFound':  
                logger.info("Parameter group %s does not exist.",  
                           parameter_group_name)  
            else:  
                logger.error(  
                    "Couldn't get parameter group %s. Here's why: %s: %s",  
                    parameter_group_name,  
                    err.response['Error']['Code'], err.response['Error']  
                    ['Message'])  
                raise  
            else:  
                return parameter_group
```

- For API details, see [DescribeDBClusterParameterGroups](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe Aurora DB cluster snapshots using an AWS SDK

The following code examples show how to describe Aurora DB cluster snapshots.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void waitForSnapshotReady(RdsClient rdsClient, String dbSnapshotIdentifier, String dbInstanceClusterIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");

        DescribeDbClusterSnapshotsRequest snapshotsRequest =
        DescribeDbClusterSnapshotsRequest.builder()
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .build();

        while (!snapshotReady) {
            DescribeDbClusterSnapshotsResponse response =
            rdsClient.describeDBClusterSnapshots(snapshotsRequest);
            List<DBClusterSnapshot> snapshotList =
            response.dbClusterSnapshots();
            for (DBClusterSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                } else {
                    System.out.println(".");
                    Thread.sleep(sleepTime * 5000);
                }
            }
        }
        System.out.println("The Snapshot is available!");
    } catch (RdsException | InterruptedException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBClusterSnapshots](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun waitSnapshotReady(dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?) {
```

```
var snapshotReady = false
var snapshotReadyStr: String
println("Waiting for the snapshot to become available.")

val snapshotsRequest = DescribeDbClusterSnapshotsRequest {
    dbClusterSnapshotIdentifier = dbSnapshotIdentifier
    dbClusterIdentifier = dbInstanceClusterIdentifier
}

RdsClient { region = "us-west-2" }.use { rdsClient ->
    while (!snapshotReady) {
        val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
        val snapshotList = response.dbClusterSnapshots
        if (snapshotList != null) {
            for (snapshot in snapshotList) {
                snapshotReadyStr = snapshot.status.toString()
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true
                } else {
                    println(".")
                    delay(slTime * 5000)
                }
            }
        }
    }
}
println("The Snapshot is available!")
}
```

- For API details, see [DescribeDBClusterSnapshots](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        
```

```
"""
try:
    response = self.rds_client.describe_db_cluster_snapshots(
        DBClusterSnapshotIdentifier=snapshot_id)
    snapshot = response['DBClusterSnapshots'][0]
except ClientError as err:
    logger.error(
        "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
        snapshot_id,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return snapshot
```

- For API details, see [DescribeDBClusterSnapshots](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe Aurora DB clusters using an AWS SDK

The following code examples show how to describe Aurora DB clusters.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDbClusterParameters(RdsClient rdsClient, String dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
                DescribeDbClusterParametersRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();
        } else {
            dbParameterGroupsRequest =
                DescribeDbClusterParametersRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .source("user")
                    .build();
        }

        DescribeDbClusterParametersResponse response =
            rdsClient.describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para: dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ( (paraName.compareTo("auto_increment_offset") ==0) ||
                (paraName.compareTo("auto_increment_increment ") ==0) ) {
                System.out.println("**** The parameter name is " + paraName);
                System.out.println("**** The parameter value is " +
                    para.parameterValue());
                System.out.println("**** The parameter data type is " +
                    para.dataType());
            }
        }
    }
}
```

```
        System.out.println("**** The parameter description is " +
para.description());
        System.out.println("**** The parameter allowed values is " +
para.allowedValues());
    }
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- For API details, see [DescribeDBClusters](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeDbClusterParameters(dbClusterGroupName: String?, flag: Int) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest = if (flag == 0) {
        DescribeDbClusterParametersRequest {
            dbClusterParameterGroupName = dbClusterGroupName
        }
    } else {
        DescribeDbClusterParametersRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            source = "user"
        }
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
        rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||

paraName.compareTo("auto_increment_increment") == 0) {
                    println("**** The parameter name is $paraName")
                    println("**** The parameter value is ${para.parameterValue}")
                    println("**** The parameter data type is ${para.dataType}")
                    println("**** The parameter description is ${para.description}")
                    println("**** The parameter allowed values is
${para.allowedValues}")
                }
            }
        }
    }
}
```

- For API details, see [DescribeDBClusters](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:  
    """Encapsulates Aurora DB cluster actions."""  
    def __init__(self, rds_client):  
        """  
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)  
        client.  
        """  
        self.rds_client = rds_client  
  
    @classmethod  
    def from_client(cls):  
        """  
        Instantiates this class from a Boto3 client.  
        """  
        rds_client = boto3.client('rds')  
        return cls(rds_client)  
  
    def get_db_cluster(self, cluster_name):  
        """  
        Gets data about an Aurora DB cluster.  
  
        :param cluster_name: The name of the DB cluster to retrieve.  
        :return: The retrieved DB cluster.  
        """  
        try:  
            response = self.rds_client.describe_db_clusters(  
                DBClusterIdentifier=cluster_name)  
            cluster = response['DBClusters'][0]  
        except ClientError as err:  
            if err.response['Error']['Code'] == 'DBClusterNotFoundFault':  
                logger.info("Cluster %s does not exist.", cluster_name)  
            else:  
                logger.error(  
                    "Couldn't verify the existence of DB cluster %s. Here's why:  
                    %s: %s", cluster_name,  
                    err.response['Error']['Code'], err.response['Error'][  
                        'Message'])  
                raise  
            else:  
                return cluster
```

- For API details, see [DescribeDBClusters](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe Aurora DB instances using an AWS SDK

The following code examples show how to describe Aurora DB instances.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String dbClusterIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");

    try {
        DescribeDbClustersRequest instanceRequest =
        DescribeDbClustersRequest.builder()
            .dbClusterIdentifier(dbClusterIdentifier)
            .build();

        while (!instanceReady) {
            DescribeDbClustersResponse response =
            rdsClient.describeDBClusters(instanceRequest);
            List<DBCluster> clusterList = response.dbClusters();
            for (DBCluster cluster : clusterList) {
                instanceReadyStr = cluster.status();
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database cluster is available!");
    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {  
    var instanceReady = false
```

```
var instanceReadyStr: String
println("Waiting for instance to become available.")
val instanceRequest = DescribeDbInstancesRequest {
    dbInstanceIdentifier = dbInstanceIdentifierVal
}

var endpoint = ""
RdsClient { region = "us-west-2" }.use { rdsClient ->
    while (!instanceReady) {
        val response = rdsClient.describeDbInstances(instanceRequest)
        response.dbInstances?.forEach { instance ->
            instanceReadyStr = instance.dbInstanceStatus.toString()
            if (instanceReadyStr.contains("available")) {
                endpoint = instance.endpoint?.address.toString()
                instanceReady = true
            } else {
                print(".")
                delay(sleepTime * 1000)
            }
        }
    }
}
println("Database instance is available! The connection endpoint is $endpoint")
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(

```

```
        DBInstanceIdentifier=instance_id)
    db_inst = response['DBInstances'][0]
except ClientError as err:
    if err.response['Error']['Code'] == 'DBInstanceNotFound':
        logger.info("Instance %s does not exist.", instance_id)
    else:
        logger.error(
            "Couldn't get DB instance %s. Here's why: %s: %s",
            instance_id,
            err.response['Error']['Code'], err.response['Error']
        )
        raise
else:
    return db_inst
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe Aurora database engine versions using an AWS SDK

The following code examples show how to describe Aurora database engine versions.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDBEngines( RdsClient rdsClient ) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
    DescribeDbEngineVersionsRequest.builder()
        .engine("aurora-mysql")
        .defaultOnly(true)
        .maxRecords(20)
        .build();

        DescribeDbEngineVersionsResponse response =
    rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engine0b: engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "+engine0b.dbParameterGroupFamily());
            System.out.println("The name of the database engine
"+engine0b.engine());
            System.out.println("The version number of the database engine
"+engine0b.engineVersion());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBEngineVersions](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest = DescribeDbEngineVersionsRequest {
        dbParameterGroupFamily = dbParameterGroupFamilyVal
        engine = "aurora-mysql"
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}
```

- For API details, see [DescribeDBEngineVersions](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.
        
```

```
:param engine: The database engine to look up.
:param parameter_group_family: When specified, restricts the returned list
of
                                engine versions to those that are compatible
with
                                this parameter group family.
:return: The list of database engine versions.
"""
try:
    kwargs = {'Engine': engine}
    if parameter_group_family is not None:
        kwargs['DBParameterGroupFamily'] = parameter_group_family
    response = self.rds_client.describe_db_engine_versions(**kwargs)
    versions = response['DBEngineVersions']
except ClientError as err:
    logger.error(
        "Couldn't get engine versions for %s. Here's why: %s: %s",
        engine,
        err.response['Error']['Code'],
        err.response['Error']['Message'])
    raise
else:
    return versions
```

- For API details, see [DescribeDBEngineVersions](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe options for Aurora DB instances using an AWS SDK

The following code examples show how to describe options for Aurora DB instances.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDBEngines( RdsClient rdsClient ) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .engine("aurora-mysql")
    .defaultOnly(true)
    .maxRecords(20)
    .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engine0b: engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "+engine0b.dbParameterGroupFamily());
            System.out.println("The name of the database engine
"+engine0b.engine());
            System.out.println("The version number of the database engine
"+engine0b.engineVersion());
        }
    }
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- For API details, see [DescribeOrderableDBInstanceOptions](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
        instance.
        :param db_engine_version: The engine version that must be supported by the
        DB instance.
        :return: The list of DB instance options that can be used to create a
        compatible DB instance.
        """
        try:
            inst_opts = []
            paginator =
                self.rds_client.get_paginator('describe_orderable_db_instance_options')
                for page in paginator.paginate(Engine=db_engine,
                EngineVersion=db_engine_version):
                    inst_opts += page['OrderableDBInstanceOptions']
            except ClientError as err:
                logger.error(
                    "Couldn't get orderable DB instances. Here's why: %s: %s",
                    err.response['Error']['Code'], err.response['Error']['Message'])
                raise
            else:
                return inst_opts
```

- For API details, see [DescribeOrderableDBInstanceOptions](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe parameters from an Aurora DB cluster parameter group using an AWS SDK

The following code examples show how to describe parameters from an Aurora DB cluster parameter group.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDbClusterParameters(RdsClient rdsClient, String dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
                DescribeDbClusterParametersRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();
        } else {
            dbParameterGroupsRequest =
                DescribeDbClusterParametersRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .source("user")
                    .build();
        }

        DescribeDbClusterParametersResponse response =
            rdsClient.describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para: dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ( (paraName.compareTo("auto_increment_offset") ==0) ||
                (paraName.compareTo("auto_increment_increment ") ==0) ) {
                System.out.println("**** The parameter name is " + paraName);
                System.out.println("**** The parameter value is " +
                    para.parameterValue());
                System.out.println("**** The parameter data type is " +
                    para.dataType());
                System.out.println("**** The parameter description is " +
                    para.description());
                System.out.println("**** The parameter allowed values is " +
                    para.allowedValues());
            }
        }
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBClusterParameters](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeDbClusterParameters(dbClusterGroupName: String?, flag: Int) {  
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest  
    dbParameterGroupsRequest = if (flag == 0) {  
        DescribeDbClusterParametersRequest {  
            dbClusterParameterGroupName = dbClusterGroupName  
        }  
    } else {  
        DescribeDbClusterParametersRequest {  
            dbClusterParameterGroupName = dbClusterGroupName  
            source = "user"  
        }  
    }  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val response =  
        rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)  
        response.parameters?.forEach { para ->  
            // Only print out information about either auto_increment_offset or  
            // auto_increment_increment.  
            val paraName = para.parameterName  
            if (paraName != null) {  
                if (paraName.compareTo("auto_increment_offset") == 0 ||  
                    paraName.compareTo("auto_increment_increment ") == 0) {  
                    println("**** The parameter name is $paraName")  
                    println("**** The parameter value is ${para.parameterValue}")  
                    println("**** The parameter data type is ${para.dataType}")  
                    println("**** The parameter description is ${para.description}")  
                    println("**** The parameter allowed values is  
                    ${para.allowedValues}")  
                }  
            }  
        }  
    }  
}
```

- For API details, see [DescribeDBClusterParameters](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_parameters(self, parameter_group_name, name_prefix='', source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
        filtered
            to contain only parameters that start with this prefix.
        :param source: When specified, only parameters from this source are
        retrieved.
            For example, a source of 'user' retrieves only parameters
        that
            were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {'DBClusterParameterGroupName': parameter_group_name}
            if source is not None:
                kwargs['Source'] = source
            parameters = []
            paginator =
self.rds_client.getPaginator('describe_db_cluster_parameters')
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p for p in page['Parameters'] if
p['ParameterName'].startswith(name_prefix)]
            except ClientError as err:
                logger.error(
                    "Couldn't get parameters for %s. Here's why: %s: %s",
parameter_group_name,
                    err.response['Error']['Code'], err.response['Error']['Message'])
                raise
            else:
                return parameters

```

- For API details, see [DescribeDBClusterParameters](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update parameters in an Aurora DB cluster parameter group using an AWS SDK

The following code examples show how to update parameters in an Aurora DB cluster parameter group.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
        .dbClusterParameterGroupName(dbClusterGroupName)
        .maxRecords(20)
        .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest).dbClusterParameterGroups();
        for (DBClusterParameterGroup group: groups) {
            System.out.println("The group name is
"+group.dbClusterParameterGroupName());
            System.out.println("The group ARN is
"+group.dbClusterParameterGroupArn());
        }
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [ModifyDBClusterParameterGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 = Parameter {
        parameterName = "auto_increment_offset"
        applyMethod = ApplyMethod.fromValue("immediate")
        parameterValue = "5"
    }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest = ModifyDbClusterParameterGroupRequest {
        dbClusterParameterGroupName = dClusterGroupName
        parameters = paraList
    }
}
```

```
        }

        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
            println("The parameter group ${response.dbClusterParameterGroupName} was
        successfully modified")
        }
    }
```

- For API details, see [ModifyDBClusterParameterGroup](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters)
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

- For API details, see [ModifyDBClusterParameterGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios for Aurora using AWS SDKs

The following code examples show how to use Amazon Aurora with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Get started with Aurora DB clusters using an AWS SDK \(p. 97\)](#)

## Get started with Aurora DB clusters using an AWS SDK

The following code examples show how to:

- Create a custom Aurora DB cluster parameter group and set parameter values.
- Create a DB cluster that is configured to use the parameter group.
- Create a DB instance in the DB cluster that contains a database.
- Take a snapshot of the DB cluster.
- Delete the instance, DB cluster, and parameter group.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
 * Before running this Java (v2) code example, set up your development environment,  
 * including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * This Java example performs the following tasks:  
 *  
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition by  
 *    calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.  
 * 2. Selects an engine family and creates a custom DB cluster parameter group by  
 *    invoking the describeDBClusterParameters method.  
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups  
 *    method.  
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters  
 *    method.  
 * 5. Modifies the auto_increment_offset parameter by invoking the  
 *    modifyDBClusterParameterGroupRequest method.  
 * 6. Gets and displays the updated parameters.  
 * 7. Gets a list of allowed engine versions by invoking the  
 *    describeDbEngineVersions method.  
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL database.  
 * 9. Waits for DB instance to be ready.  
 * 10. Gets a list of instance classes available for the selected engine.  
 * 11. Creates a database instance in the cluster.  
 * 12. Waits for DB instance to be ready.  
 * 13. Creates a snapshot.  
 * 14. Waits for DB snapshot to be ready.  
 * 15. Deletes the DB cluster.
```

```
* 16. Deletes the DB cluster group.  
*/  
  
public class AuroraScenario {  
    public static long sleepTime = 20;  
    public static final String DASHES = new String(new char[80]).replace("\0",  
"-");  
    public static void main(String[] args) throws InterruptedException {  
  
        final String usage = "\n" +  
            "Usage:\n" +  
            "  <dbClusterGroupName> <dbParameterGroupFamily>  
<dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>  
<dbSnapshotIdentifier> <username> <userPassword>" +  
            "Where:\n" +  
            "  dbClusterGroupName - The name of the DB cluster parameter group.  
\n" +  
            "  dbParameterGroupFamily - The DB cluster parameter group family  
name (for example, aurora-mysql5.7). \n"+  
            "  dbInstanceClusterIdentifier - The instance cluster identifier  
value.\n"+  
            "  dbInstanceIdentifier - The database instance identifier.\n"+  
            "  dbName  The database name.\n"+  
            "  dbSnapshotIdentifier - The snapshot identifier.\n"+  
            "  username - The database user name.\n" +  
            "  userPassword - The database user name password.\n" ;  
  
        if (args.length != 8) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String dbClusterGroupName = args[0];  
        String dbParameterGroupFamily = args[1];  
        String dbInstanceClusterIdentifier = args[2];  
        String dbInstanceIdentifier = args[3];  
        String dbName = args[4];  
        String dbSnapshotIdentifier = args[5];  
        String username = args[6];  
        String userPassword = args[7];  
  
        Region region = Region.US_WEST_2;  
        RdsClient rdsClient = RdsClient.builder()  
            .region(region)  
            .credentialsProvider(ProfileCredentialsProvider.create())  
            .build();  
  
        System.out.println(DASHES);  
        System.out.println("Welcome to the Amazon Aurora example scenario.");  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("1. Return a list of the available DB engines");  
        describeDBEngines(rdsClient);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("2. Create a custom parameter group");  
        createDBClusterParameterGroup(rdsClient, dbClusterGroupName,  
        dbParameterGroupFamily);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("3. Get the parameter group");  
        describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);  
        System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("4. Get the parameters in the group");
describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Modify the auto_increment_offset parameter");
modifyDBClusterParas(rdsClient, dbClusterGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Display the updated parameter value");
describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of allowed engine versions");
getAllowedEngines(rdsClient, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Create an Aurora DB cluster database");
String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier, username, userPassword) ;
System.out.println("The ARN of the cluster is "+arnClusterVal);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Wait for DB instance to be ready" );
waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a list of instance classes available for the
selected engine");
String instanceClass = getListInstanceClasses(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Create a database instance in the cluster.");
String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier, instanceClass);
System.out.println("The ARN of the database is "+clusterDBARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Wait for DB instance to be ready" );
waitForDBInstanceReady(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Create a snapshot");
createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Wait for DB snapshot to be ready" );
waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Delete the DB instance" );
deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Delete the DB cluster");
deleteCluster(rdsClient, dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Delete the DB cluster group");
deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The Scenario has successfully completed." );
System.out.println(DASHES);
rdsClient.close();
}

public static void deleteDBClusterGroup( RdsClient rdsClient, String dbClusterGroupName, String clusterDBARN) throws InterruptedException {
try {
    boolean isDataDel = false;
    boolean didFind;
    String instanceARN ;

    // Make sure that the database has been deleted.
    while (!isDataDel) {
        DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
        List<DBInstance> instanceList = response.dbInstances();
        int listSize = instanceList.size();
        isDataDel = false ;
        didFind = false;
        int index = 1;
        for (DBInstance instance: instanceList) {
            instanceARN = instance.dbInstanceArn();
            if (instanceARN.compareTo(clusterDBARN) == 0) {
                System.out.println(clusterDBARN + " still exists");
                didFind = true ;
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
database ARN.
                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }
    DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest.builder()
    .dbClusterParameterGroupName(dbClusterGroupName)
    .build();

    rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
    System.out.println(dbClusterGroupName +" was deleted.");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

public static void deleteCluster(RdsClient rdsClient, String dbInstanceClusterIdentifier) {
```

```
        try {
            DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                .dbClusterIdentifier(dbInstanceClusterIdentifier)
                .skipFinalSnapshot(true)
                .build();

            rdsClient.deleteDBCluster(deleteDbClusterRequest);
            System.out.println(dbInstanceClusterIdentifier +" was deleted!");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void deleteDatabaseInstance( RdsClient rdsClient, String dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .deleteAutomatedBackups(true)
                .skipFinalSnapshot(true)
                .build();

            DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.println("The status of the database is " +
response.dbInstance().dbInstanceState());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitForSnapshotReady(RdsClient rdsClient, String dbSnapshotIdentifier, String dbInstanceClusterIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                .dbClusterIdentifier(dbInstanceClusterIdentifier)
                .build();

            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots (snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }
        }
    }
}
```

```
        System.out.println("The Snapshot is available!");

    } catch (RdsException | InterruptedException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void createDBClusterSnapshot(RdsClient rdsClient, String dbInstanceClusterIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
        .build();

        CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
        System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void waitDBInstanceReady(RdsClient rdsClient, String dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");

    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .build();

        String endpoint="";
        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceState();
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint().address();
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database instance is available! The connection
endpoint is "+ endpoint);

    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public static String createDBInstanceCluster(RdsClient rdsClient,
                                             String dbInstanceIdentifier,
                                             String dbInstanceClusterIdentifier,
                                             String instanceClass){

    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .dbClusterIdentifier(dbInstanceClusterIdentifier)
        .engine("aurora-mysql")
        .dbInstanceClass(instanceClass)
        .build() ;

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceState());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static String getListInstanceClasses(RdsClient rdsClient) {
    try{
        DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest.builder()
        .engine("aurora-mysql")
        .maxRecords(20)
        .build();

        DescribeOrderableDbInstanceOptionsResponse response =
rdsClient.describeOrderableDBInstanceOptions(optionsRequest);
        List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
        String instanceClass = "";
        for (OrderableDBInstanceOption instanceOption: instanceOptions) {
            instanceClass = instanceOption.dbInstanceClass();
            System.out.println("The instance class is "
+instanceOption.dbInstanceClass());
            System.out.println("The engine version is "
+instanceOption.engineVersion());
        }
        return instanceClass;

    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");

    try {
```

```
DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
    .dbClusterIdentifier(dbClusterIdentifier)
    .build();

        while (!instanceReady) {
            DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
            List<DBCluster> clusterList = response.dbClusters();
            for (DBCluster cluster : clusterList) {
                instanceReadyStr = cluster.status();
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database cluster is available!");

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName, String dbClusterIdentifier, String
userName, String password) {
    try {
        CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
    .databaseName(dbName)
    .dbClusterIdentifier(dbClusterIdentifier)
    .dbClusterParameterGroupName(dbParameterGroupFamily)
    .engine("aurora-mysql")
    .masterUsername(userName)
    .masterUserPassword(password)
    .build();

        CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
        return response.dbCluster().dbClusterArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .dbParameterGroupFamily(dbParameterGroupFamily)
    .engine("aurora-mysql")
    .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
```

```

        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine: dbEngines) {
            System.out.println("The engine version is "
+dbEngine.engineVersion());
            System.out.println("The engine description is "
+dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Modify the auto_increment_offset parameter.
public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dClusterGroupName)
            .parameters(paraList)
            .build();

        ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
        System.out.println("The parameter group "+
response.dbClusterParameterGroupName() +" was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbCLusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbCLusterGroupName)
                .source("user")
                .build();
        }

        DescribeDbClusterParametersResponse response =
rdsClient.describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para: dbParameters) {

```

```
// Only print out information about either auto_increment_offset or
auto_increment_increment.
    paraName = para.parameterName();
    if ( (paraName.compareTo("auto_increment_offset") ==0) ||
(paraName.compareTo("auto_increment_increment ") ==0)) {
        System.out.println("**** The parameter name is " + paraName);
        System.out.println("**** The parameter value is " +
para.parameterValue());
        System.out.println("**** The parameter data type is " +
para.dataType());
        System.out.println("**** The parameter description is " +
para.description());
        System.out.println("**** The parameter allowed values is " +
para.allowedValues());
    }
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
        .dbClusterParameterGroupName(dbClusterGroupName)
        .maxRecords(20)
        .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest).dbClusterParameterGroups();
        for (DBClusterParameterGroup group: groups) {
            System.out.println("The group name is
"+group.dbClusterParameterGroupName());
            System.out.println("The group ARN is
"+group.dbClusterParameterGroupArn());
        }
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
        .dbClusterParameterGroupName(dbClusterGroupName)
        .dbParameterGroupFamily(dbParameterGroupFamily)
        .description("Created by using the AWS SDK for Java")
        .build();

        CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
        System.out.println("The group name is "+
response.dbClusterParameterGroup().dbClusterParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
}

public static void describeDBEngines( RdsClient rdsClient ) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .engine("aurora-mysql")
    .defaultOnly(true)
    .maxRecords(20)
    .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engine0b: engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "+engine0b.dbParameterGroupFamily());
            System.out.println("The name of the database engine
"+engine0b.engine());
            System.out.println("The version number of the database engine
"+engine0b.engineVersion());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [CreateDBCluster](#)
  - [CreateDBClusterParameterGroup](#)
  - [CreateDBClusterSnapshot](#)
  - [CreateDBInstance](#)
  - [DeleteDBCluster](#)
  - [DeleteDBClusterParameterGroup](#)
  - [DeleteDBInstance](#)
  - [DescribeDBClusterParameterGroups](#)
  - [DescribeDBClusterParameters](#)
  - [DescribeDBClusterSnapshots](#)
  - [DescribeDBClusters](#)
  - [DescribeDBEngineVersions](#)
  - [DescribeDBInstances](#)
  - [DescribeOrderableDBInstanceOptions](#)
  - [ModifyDBClusterParameterGroup](#)

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
Before running this Kotlin code example, set up your development environment,  
including your credentials.  
  
For more information, see the following documentation topic:  
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html  
  
This Kotlin example performs the following tasks:  
  
1. Returns a list of the available DB engines.  
2. Creates a custom DB parameter group.  
3. Gets the parameter groups.  
4. Gets the parameters in the group.  
5. Modifies the auto_increment_increment parameter.  
6. Displays the updated parameter value.  
7. Gets a list of allowed engine versions.  
8. Creates an Aurora DB cluster database.  
9. Waits for DB instance to be ready.  
10. Gets a list of instance classes available for the selected engine.  
11. Creates a database instance in the cluster.  
12. Waits for the database instance in the cluster to be ready.  
13. Creates a snapshot.  
14. Waits for DB snapshot to be ready.  
15. Deletes the DB instance.  
16. Deletes the DB cluster.  
17. Deletes the DB cluster group.  
*/  
  
var slTime: Long = 20  
suspend fun main(args: Array<String>) {  
    val usage = """  
        Usage:  
        <dbClusterGroupName> <dbParameterGroupFamily>  
        <dbInstanceClusterIdentifier> <dbName> <dbSnapshotIdentifier> <username>  
        <userPassword>  
  
        Where:  
        dbClusterGroupName - The database group name.  
        dbParameterGroupFamily - The database parameter group name.  
        dbInstanceClusterIdentifier - The database instance identifier.  
        dbName - The database name.  
        dbSnapshotIdentifier - The snapshot identifier.  
        username - The user name.  
        userPassword - The password that corresponds to the user name.  
    """  
  
    if (args.size != 7) {  
        println(usage)  
        exitProcess(1)  
    }  
  
    val dbClusterGroupName = args[0]  
    val dbParameterGroupFamily = args[1]  
    val dbInstanceClusterIdentifier = args[2]  
    val dbInstanceIdentifier = args[3]  
    val dbName = args[4]  
    val dbSnapshotIdentifier = args[5]  
    val username = args[6]  
    val userPassword = args[7]
```

```
println("1. Return a list of the available DB engines")
describeAuroraDBEngines()

println("2. Create a custom parameter group")
createDBClusterParameterGroup(dbClusterGroupName, dbParameterGroupFamily)

println("3. Get the parameter group")
describeDbClusterParameterGroups(dbClusterGroupName)

println("4. Get the parameters in the group")
describeDbClusterParameters(dbClusterGroupName, 0)

println("5. Modify the auto_increment_offset parameter")
modifyDBClusterParas(dbClusterGroupName)

println("6. Display the updated parameter value")
describeDbClusterParameters(dbClusterGroupName, -1)

println("7. Get a list of allowed engine versions")
getAllowedClusterEngines(dbParameterGroupFamily)

println("8. Create an Aurora DB cluster database")
val arnClusterVal = createDBCluster(dbClusterGroupName, dbName,
dbInstanceClusterIdentifier, username, userPassword)
println("The ARN of the cluster is $arnClusterVal")

println("9. Wait for DB instance to be ready")
waitForClusterInstanceReady(dbInstanceClusterIdentifier)

println("10. Get a list of instance classes available for the selected engine")
val instanceClass = getListInstanceClasses()

println("11. Create a database instance in the cluster.")
val clusterDBARN = createDBInstanceCluster(dbInstanceIdentifier,
dbInstanceClusterIdentifier, instanceClass)
println("The ARN of the database is $clusterDBARN")

println("12. Wait for DB instance to be ready")
waitForAuroraInstanceReady(dbInstanceIdentifier)

println("13. Create a snapshot")
createDBClusterSnapshot(dbInstanceClusterIdentifier, dbSnapshotIdentifier)

println("14. Wait for DB snapshot to be ready")
waitForSnapshotReady(dbSnapshotIdentifier, dbInstanceClusterIdentifier)

println("15. Delete the DB instance")
deleteDBInstance(dbInstanceIdentifier)

println("16. Delete the DB cluster")
deleteCluster(dbInstanceClusterIdentifier)

println("17. Delete the DB cluster group")
if (clusterDBARN != null) {
    deleteDBClusterGroup(dbClusterGroupName, clusterDBARN)
}
println("The Scenario has successfully completed.")

}

@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(dbClusterGroupName: String, clusterDBARN: String)
{
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String
```

```
RdsClient { region = "us-west-2" }.use { rdsClient ->
    // Make sure that the database has been deleted.
    while (!isDataDel) {
        val response = rdsClient.describeDbInstances()
        val instanceList = response.dbInstances
        val listSize = instanceList?.size
        isDataDel = false
        didFind = false
        var index = 1
        if (instanceList != null) {
            for (instance in instanceList) {
                instanceARN = instance.dbInstanceArn.toString()
                if (instanceARN.compareTo(clusterDBARN) == 0) {
                    println("$clusterDBARN still exists")
                    didFind = true
                }
                if (index == listSize && !didFind) {
                    // Went through the entire list and did not find the
                    database ARN.
                    isDataDel = true
                }
                delay(slTime * 1000)
                index++
            }
        }
    }
    val clusterParameterGroupRequest = DeleteDbClusterParameterGroupRequest {
        dbClusterParameterGroupName = dbClusterGroupName
    }

    rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
    println("$dbClusterGroupName was deleted.")
}
}

suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest = DeleteDbClusterRequest {
        dbClusterIdentifier = dbInstanceClusterIdentifier
        skipFinalSnapshot = true
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}

suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest = DeleteDbInstanceRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
        deleteAutomatedBackups = true
        skipFinalSnapshot = true
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
        ${response.dbInstance?.dbInstanceState}")
    }
}

suspend fun waitSnapshotReady(dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?) {
    var snapshotReady = false
    var snapshotReadyStr: String
```

```
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest = DescribeDbClusterSnapshotsRequest {
        dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        dbClusterIdentifier = dbInstanceClusterIdentifier
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

suspend fun createDBClusterSnapshot(dbInstanceClusterIdentifier: String?,
                                    dbSnapshotIdentifier: String?) {
    val snapshotRequest = CreateDbClusterSnapshotRequest {
        dbClusterIdentifier = dbInstanceClusterIdentifier
        dbClusterSnapshotIdentifier = dbSnapshotIdentifier
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}

suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest = DescribeDbInstancesRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
    }

    var endpoint = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceState.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}
```

```

}

suspend fun createDBInstanceCluster(dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?, instanceClassVal: String?): String? {
    val instanceRequest = CreateDbInstanceRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
        dbClusterIdentifier = dbInstanceClusterIdentifierVal
        engine = "aurora-mysql"
        dbInstanceClass = instanceClassVal
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceState}")
        return response.dbInstance?.dbInstanceArn
    }
}

suspend fun getListInstanceClasses(): String {
    val optionsRequest = DescribeOrderableDbInstanceOptionsRequest {
        engine = "aurora-mysql"
        maxRecords = 20
    }
    var instanceClass = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
            instanceClass = instanceOption.dbInstanceClass.toString()
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
    return instanceClass
}

// Waits until the database instance is available.
suspend fun waitForClusterInstanceReady(dbClusterIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest = DescribeDbClustersRequest {
        dbClusterIdentifier = dbClusterIdentifierVal
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbClusters(instanceRequest)
            response.dbClusters?.forEach { cluster ->
                instanceReadyStr = cluster.status.toString()
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database cluster is available!")
}

suspend fun createDBCluster(dbParameterGroupFamilyVal: String?, dbName: String?,
    dbClusterIdentifierVal: String?, userName: String?, password: String?): String? {
    val clusterRequest = CreateDbClusterRequest {
        databaseName = dbName

```

```
        dbClusterIdentifier = dbClusterIdentifierVal
        dbClusterParameterGroupName = dbParameterGroupFamilyVal
        engine = "aurora-mysql"
        masterUsername = userName
        masterUserPassword = password
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest = DescribeDbEngineVersionsRequest {
        dbParameterGroupFamily = dbParameterGroupFamilyVal
        engine = "aurora-mysql"
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 = Parameter {
        parameterName = "auto_increment_offset"
        applyMethod = ApplyMethod.fromValue("immediate")
        parameterValue = "5"
    }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest = ModifyDbClusterParameterGroupRequest {
        dbClusterParameterGroupName = dClusterGroupName
        parameters = paraList
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
successfully modified")
    }
}

suspend fun describeDbClusterParameters(dbClusterGroupName: String?, flag: Int) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest = if (flag == 0) {
        DescribeDbClusterParametersRequest {
            dbClusterParameterGroupName = dbClusterGroupName
        }
    } else {
        DescribeDbClusterParametersRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            source = "user"
        }
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
```

```

        val response =
    rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 || paraName.compareTo("auto_increment_increment") == 0) {
                    println("**** The parameter name is $paraName")
                    println("**** The parameter value is ${para.parameterValue}")
                    println("**** The parameter data type is ${para.dataType}")
                    println("**** The parameter description is ${para.description}")
                    println("**** The parameter allowed values is
${para.allowedValues}")
                }
            }
        }
    }

suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest = DescribeDbClusterParameterGroupsRequest {
        dbClusterParameterGroupName = dbClusterGroupName
        maxRecords = 20
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}

suspend fun createDBClusterParameterGroup(dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?) {
    val groupRequest = CreateDbClusterParameterGroupRequest {
        dbClusterParameterGroupName = dbClusterGroupNameVal
        dbParameterGroupFamily = dbParameterGroupFamilyVal
        description = "Created by using the AWS SDK for Kotlin"
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}

suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest = DescribeDbEngineVersionsRequest {
        engine = "aurora-mysql"
        defaultOnly = true
        maxRecords = 20
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engine0b ->
            println("The name of the DB parameter group family for the database
engine is ${engine0b.dbParameterGroupFamily}")
            println("The name of the database engine ${engine0b.engine}")
            println("The version number of the database engine
${engine0b.engineVersion}")
        }
    }
}

```

```
        }  
    }  
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - [CreateDBCluster](#)
  - [CreateDBClusterParameterGroup](#)
  - [CreateDBClusterSnapshot](#)
  - [CreateDBInstance](#)
  - [DeleteDBCluster](#)
  - [DeleteDBClusterParameterGroup](#)
  - [DeleteDBInstance](#)
  - [DescribeDBClusterParameterGroups](#)
  - [DescribeDBClusterParameters](#)
  - [DescribeDBClusterSnapshots](#)
  - [DescribeDBClusters](#)
  - [DescribeDBEngineVersions](#)
  - [DescribeDBInstances](#)
  - [DescribeOrderableDBInstanceOptions](#)
  - [ModifyDBClusterParameterGroup](#)

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
class AuroraClusterScenario:  
    """Runs a scenario that shows how to get started using Aurora DB clusters."""  
    def __init__(self, aurora_wrapper):  
        """  
        :param aurora_wrapper: An object that wraps Aurora DB cluster actions.  
        """  
        self.aurora_wrapper = aurora_wrapper  
  
    def create_parameter_group(self, db_engine, parameter_group_name):  
        """  
        Shows how to get available engine versions for a specified database engine  
        and  
        create a DB cluster parameter group that is compatible with a selected  
        engine family.  
  
        :param db_engine: The database engine to use as a basis.  
        :param parameter_group_name: The name given to the newly created parameter  
        group.  
        :return: The newly created parameter group.  
        """  
        print(f"Checking for an existing DB cluster parameter group named  
{parameter_group_name}.")  
        parameter_group =  
        self.aurora_wrapper.get_parameter_group(parameter_group_name)  
        if parameter_group is None:
```

```
        print(f"Getting available database engine versions for {db_engine}.")
        engine_versions = self.aurora_wrapper.get_engine_versions(db_engine)
        families = list({ver['DBParameterGroupFamily'] for ver in
            engine_versions})
        family_index = q.choose("Which family do you want to use? ", families)
        print(f"Creating a DB cluster parameter group.")
        self.aurora_wrapper.create_parameter_group(
            parameter_group_name, families[family_index], 'Example parameter
group.')
        parameter_group =
self.aurora_wrapper.get_parameter_group(parameter_group_name)
        print(f"Parameter group {parameter_group['DBClusterParameterGroupName']}:")
        pp(parameter_group)
        print('*'*88)
        return parameter_group

def set_user_parameters(self, parameter_group_name):
    """
    Shows how to get the parameters contained in a custom parameter group and
    update some of the parameter values in the group.

    :param parameter_group_name: The name of the parameter group to query and
    modify.
    """
    print("Let's set some parameter values in your parameter group.")
    auto_inc_parameters = self.aurora_wrapper.get_parameters(
        parameter_group_name, name_prefix='auto_increment')
    update_params = []
    for auto_inc in auto_inc_parameters:
        if auto_inc['IsModifiable'] and auto_inc['DataType'] == 'integer':
            print(f"The {auto_inc['ParameterName']} parameter is described
as:")
            print(f"\t{auto_inc['Description']}")
            param_range = auto_inc['AllowedValues'].split('-')
            auto_inc['ParameterValue'] = str(q.ask(
                f"Enter a value between {param_range[0]} and {param_range[1]}:
",
                q.is_int, q.in_range(int(param_range[0]),
int(param_range[1]))))
            update_params.append(auto_inc)
    self.aurora_wrapper.update_parameters(parameter_group_name, update_params)
    print("You can get a list of parameters you've set by specifying a source
of 'user'.")
    user_parameters = self.aurora_wrapper.get_parameters(parameter_group_name,
source='user')
    pp(user_parameters)
    print('*'*88)

def create_cluster(self, cluster_name, db_engine, db_name, parameter_group):
    """
    Shows how to create an Aurora DB cluster that contains a database of a
specified
    type. The database is also configured to use a custom DB cluster parameter
group.

    :param cluster_name: The name given to the newly created DB cluster.
    :param db_engine: The engine of the created database.
    :param db_name: The name given to the created database.
    :param parameter_group: The parameter group that is associated with the DB
cluster.
    :return: The newly created DB cluster.
    """
    print("Checking for an existing DB cluster.")
    cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
    if cluster is None:
        admin_username = q.ask(
```

```

        "Enter an administrator user name for the database: ", q.non_empty)
    admin_password = q.ask(
        "Enter a password for the administrator (at least 8 characters): ",
    q.non_empty)
    engine_versions = self.aurora_wrapper.get_engine_versions(
        db_engine, parameter_group['DBParameterGroupFamily'])
    engine_choices = [ver['EngineVersion'] for ver in engine_versions]
    print("The available engines for your parameter group are:")
    engine_index = q.choose("Which engine do you want to use? ",
    engine_choices)
    print(f"Creating DB cluster {cluster_name} and database {db_name}.\\n"
        f"The DB cluster is configured to use\\n"
        f"your custom parameter group
{parameter_group['DBClusterParameterGroupName']}\\n"
        f"and selected engine {engine_choices[engine_index]}.\\n"
        f"This typically takes several minutes.")
    cluster = self.aurora_wrapper.create_db_cluster(
        cluster_name, parameter_group['DBClusterParameterGroupName'],
    db_name,
        db_engine, engine_choices[engine_index], admin_username,
    admin_password)
    while cluster.get('Status') != 'available':
        wait(10)
        cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
    print("Cluster created and available.\\n")
    print("Cluster data:")
    pp(cluster)
    print('*'*88)
    return cluster

def create_instance(self, cluster):
    """
    Shows how to create a DB instance in an existing Aurora DB cluster. A new
    DB cluster
        contains no DB instances, so you must add one. The first DB instance that
    is added
        to a DB cluster defaults to a read-write DB instance.

    :param cluster: The DB cluster where the DB instance is added.
    :return: The newly created DB instance.
    """
    print("Checking for an existing database instance.")
    cluster_name = cluster['DBClusterIdentifier']
    db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
    if db_inst is None:
        print("Let's create a database instance in your DB cluster.")
        print("First, choose a DB instance type:")
        inst_opts = self.aurora_wrapper.get_orderable_instances(
            cluster['Engine'], cluster['EngineVersion'])
        inst_choices = list({opt['DBInstanceClass'] for opt in inst_opts})
        inst_index = q.choose("Which DB instance class do you want to use? ",
    inst_choices)
        print(f"Creating a database instance. This typically takes several
minutes.")
        db_inst = self.aurora_wrapper.create_instance_in_cluster(
            cluster_name, cluster_name, cluster['Engine'],
    inst_choices[inst_index])
        while db_inst.get('DBInstanceState') != 'available':
            wait(10)
            db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
    print("Instance data:")
    pp(db_inst)
    print('*'*88)
    return db_inst

@staticmethod

```

```

def display_connection(cluster):
    """
    Displays connection information about an Aurora DB cluster and tips on how
    to
    connect to it.

    :param cluster: The DB cluster to display.
    """
    print("You can now connect to your database using your favorite MySql
client.\n"
          "One way to connect is by using the 'mysql' shell on an Amazon EC2
instance\n"
          "that is running in the same VPC as your database cluster. Pass the
endpoint,\n"
          "port, and administrator user name to 'mysql' and enter your password
\n"
          "when prompted:\n")
    print(f"\n\tmysql -h {cluster['Endpoint']} -P {cluster['Port']} -u
{cluster['MasterUsername']} -p\n")
    print("For more information, see the User Guide for Aurora:\n"
          "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora.Conne
print('*88)

def create_snapshot(self, cluster_name):
    """
    Shows how to create a DB cluster snapshot and wait until it's available.

    :param cluster_name: The name of a DB cluster to snapshot.
    """
    if q.ask("Do you want to create a snapshot of your DB cluster (y/n)? ",
q.is_yesno):
        snapshot_id = f"{cluster_name}-{uuid.uuid4()}"
        print(f"Creating a snapshot named {snapshot_id}. This typically takes a
few minutes.")
        snapshot = self.aurora_wrapper.create_cluster_snapshot(snapshot_id,
cluster_name)
        while snapshot.get('Status') != 'available':
            wait(10)
            snapshot = self.aurora_wrapper.get_cluster_snapshot(snapshot_id)
        pp(snapshot)
        print('*88)

def cleanup(self, db_inst, cluster, parameter_group):
    """
    Shows how to clean up a DB instance, DB cluster, and DB cluster parameter
group.
    Before the DB cluster parameter group can be deleted, all associated DB
instances and
    DB clusters must first be deleted.

    :param db_inst: The DB instance to delete.
    :param cluster: The DB cluster to delete.
    :param parameter_group: The DB cluster parameter group to delete.
    """
    cluster_name = cluster['DBClusterIdentifier']
    parameter_group_name = parameter_group['DBClusterParameterGroupName']
    if q.ask(
          "\nDo you want to delete the database instance, DB cluster, and
parameter "
          "group (y/n)? ", q.is_yesno):
        print(f"Deleting database instance {db_inst['DBInstanceIdentifier'].}")
        self.aurora_wrapper.delete_db_instance(db_inst['DBInstanceIdentifier'])
        print(f"Deleting database cluster {cluster_name}.")
        self.aurora_wrapper.delete_db_cluster(cluster_name)
        print("Waiting for the DB instance and DB cluster to delete.\n")

```

```

        "This typically takes several minutes.")
    while db_inst is not None or cluster is not None:
        wait(10)
        if db_inst is not None:
            db_inst =
self.aurora_wrapper.get_db_instance(db_inst['DBInstanceIdentifier'])
            if cluster is not None:
                cluster =
self.aurora_wrapper.get_db_cluster(cluster['DBClusterIdentifier'])
                print(f"Deleting parameter group {parameter_group_name}.")
                self.aurora_wrapper.delete_parameter_group(parameter_group_name)

def run_scenario(self, db_engine, parameter_group_name, cluster_name, db_name):
    print('*'*88)
    print("Welcome to the Amazon Relational Database Service (Amazon RDS) get
started\n"
        "with Aurora DB clusters demo.")
    print('*'*88)

    parameter_group = self.create_parameter_group(db_engine,
parameter_group_name)
        self.set_user_parameters(parameter_group_name)
        cluster = self.create_cluster(cluster_name, db_engine, db_name,
parameter_group)
        db_inst = self.create_instance(cluster)
        self.display_connection(cluster)
        self.create_snapshot(cluster_name)
        self.cleanup(db_inst, cluster, parameter_group)

    print("\nThanks for watching!")
    print('*'*88)

if __name__ == '__main__':
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    try:
        scenario = AuroraClusterScenario(AuroraWrapper.from_client())
        scenario.run_scenario(
            'aurora-mysql', 'doc-example-cluster-parameter-group', 'doc-example-
aurora',
            'docexampledb')
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Define functions that are called by the scenario to manage Aurora actions.

```

class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_parameter_group(self, parameter_group_name):
```

```
"""
Gets a DB cluster parameter group.

:param parameter_group_name: The name of the parameter group to retrieve.
:return: The requested parameter group.
"""
try:
    response = self.rds_client.describe_db_cluster_parameter_groups(
        DBClusterParameterGroupName=parameter_group_name)
    parameter_group = response['DBClusterParameterGroups'][0]
except ClientError as err:
    if err.response['Error']['Code'] == 'DBParameterGroupNotFound':
        logger.info("Parameter group %s does not exist.", parameter_group_name)
    else:
        logger.error(
            "Couldn't get parameter group %s. Here's why: %s: %s",
            parameter_group_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
else:
    return parameter_group

def create_parameter_group(self, parameter_group_name, parameter_group_family,
                           description):
    """
    Creates a DB cluster parameter group that is based on the specified
    parameter group
    family.

    :param parameter_group_name: The name of the newly created parameter group.
    :param parameter_group_family: The family that is used as the basis of the
                                   new
                                   parameter group.
    :param description: A description given to the parameter group.
    :return: Data about the newly created parameter group.
    """
    try:
        response = self.rds_client.create_db_cluster_parameter_group(
            DBClusterParameterGroupName=parameter_group_name,
            DBParameterGroupFamily=parameter_group_family,
            Description=description)
    except ClientError as err:
        logger.error(
            "Couldn't create parameter group %s. Here's why: %s: %s",
            parameter_group_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response

def delete_parameter_group(self, parameter_group_name):
    """
    Deletes a DB cluster parameter group.

    :param parameter_group_name: The name of the parameter group to delete.
    :return: Data about the parameter group.
    """
    try:
        response = self.rds_client.delete_db_cluster_parameter_group(
            DBClusterParameterGroupName=parameter_group_name)
    except ClientError as err:
        logger.error(
            "Couldn't delete parameter group %s. Here's why: %s: %s",
            parameter_group_name,
```

```
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response

def get_parameters(self, parameter_group_name, name_prefix='', source=None):
    """
    Gets the parameters that are contained in a DB cluster parameter group.

    :param parameter_group_name: The name of the parameter group to query.
    :param name_prefix: When specified, the retrieved list of parameters is
    filtered
        to contain only parameters that start with this prefix.
    :param source: When specified, only parameters from this source are
    retrieved.
        For example, a source of 'user' retrieves only parameters
    that
        were set by a user.
    :return: The list of requested parameters.
    """
    try:
        kwargs = {'DBClusterParameterGroupName': parameter_group_name}
        if source is not None:
            kwargs['Source'] = source
        parameters = []
        paginator =
self.rds_client.getPaginator('describe_db_cluster_parameters')
        for page in paginator.paginate(**kwargs):
            parameters += [
                p for p in page['Parameters'] if
p['ParameterName'].startswith(name_prefix)]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
    else:
        return parameters

def update_parameters(self, parameter_group_name, update_parameters):
    """
    Updates parameters in a custom DB cluster parameter group.

    :param parameter_group_name: The name of the parameter group to update.
    :param update_parameters: The parameters to update in the group.
    :return: Data about the modified parameter group.
    """
    try:
        response = self.rds_client.modify_db_cluster_parameter_group(
            DBClusterParameterGroupName=parameter_group_name,
            Parameters=update_parameters)
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
    else:
        return response

def get_db_cluster(self, cluster_name):
    """
    Gets data about an Aurora DB cluster.

    :param cluster_name: The name of the DB cluster to retrieve.
    
```

```
:return: The retrieved DB cluster.  
"""  
try:  
    response = self.rds_client.describe_db_clusters(  
        DBClusterIdentifier=cluster_name)  
    cluster = response['DBClusters'][0]  
except ClientError as err:  
    if err.response['Error']['Code'] == 'DBClusterNotFoundFault':  
        logger.info("Cluster %s does not exist.", cluster_name)  
    else:  
        logger.error(  
            "Couldn't verify the existence of DB cluster %s. Here's why:  
%s: %s", cluster_name,  
            err.response['Error']['Code'], err.response['Error']  
['Message'])  
        raise  
else:  
    return cluster  
  
def create_db_cluster(  
    self, cluster_name, parameter_group_name, db_name, db_engine,  
    db_engine_version, admin_name, admin_password):  
    """  
    Creates a DB cluster that is configured to use the specified parameter  
group.  
    The newly created DB cluster contains a database that uses the specified  
engine and  
engine version.  
  
    :param cluster_name: The name of the DB cluster to create.  
    :param parameter_group_name: The name of the parameter group to associate  
with  
                    the DB cluster.  
    :param db_name: The name of the database to create.  
    :param db_engine: The database engine of the database that is created, such  
as MySql.  
    :param db_engine_version: The version of the database engine.  
    :param admin_name: The user name of the database administrator.  
    :param admin_password: The password of the database administrator.  
    :return: The newly created DB cluster.  
    """  
    try:  
        response = self.rds_client.create_db_cluster(  
            DatabaseName=db_name,  
            DBClusterIdentifier=cluster_name,  
            DBClusterParameterGroupName=parameter_group_name,  
            Engine=db_engine,  
            EngineVersion=db_engine_version,  
            MasterUsername=admin_name,  
            MasterUserPassword=admin_password)  
        cluster = response['DBCluster']  
    except ClientError as err:  
        logger.error(  
            "Couldn't create database %s. Here's why: %s: %s", db_name,  
            err.response['Error']['Code'], err.response['Error']['Message'])  
        raise  
    else:  
        return cluster  
  
def delete_db_cluster(self, cluster_name):  
    """  
    Deletes a DB cluster.  
  
    :param cluster_name: The name of the DB cluster to delete.  
    """  
    try:        pass
```

```
        self.rds_client.delete_db_cluster(
            DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True)
        logger.info("Deleted DB cluster %s.", cluster_name)
    except ClientError:
        logger.exception("Couldn't delete DB cluster %s.", cluster_name)
        raise

    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
                DBClusterIdentifier=cluster_id)
            snapshot = response['DBClusterSnapshot']
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id, err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return snapshot

    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id)
            snapshot = response['DBClusterSnapshots'][0]
        except ClientError as err:
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id, err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return snapshot

    def create_instance_in_cluster(self, instance_id, cluster_id, db_engine,
                                  instance_class):
        """
        Creates a database instance in an existing DB cluster. The first database
        that is created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
                           created.
        :param db_engine: The database engine of a database to create in the DB
                          instance.
                                         This must be compatible with the configured parameter
        group
                                         of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
                              instance.
        :return: Data about the newly created DB instance.
        """

```

```

"""
try:
    response = self.rds_client.create_db_instance(
        DBInstanceIdentifier=instance_id, DBClusterIdentifier=cluster_id,
        Engine=db_engine, DBInstanceClass=instance_class)
    db_inst = response['DBInstance']
except ClientError as err:
    logger.error(
        "Couldn't create DB instance %s. Here's why: %s: %s",
        instance_id, err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return db_inst

def get_engine_versions(self, engine, parameter_group_family=None):
    """
    Gets database engine versions that are available for the specified engine
    and parameter group family.

    :param engine: The database engine to look up.
    :param parameter_group_family: When specified, restricts the returned list
        of engine versions to those that are compatible
        with this parameter group family.
    :return: The list of database engine versions.
    """
    try:
        kwargs = {'Engine': engine}
        if parameter_group_family is not None:
            kwargs['DBParameterGroupFamily'] = parameter_group_family
        response = self.rds_client.describe_db_engine_versions(**kwargs)
        versions = response['DBEngineVersions']
    except ClientError as err:
        logger.error(
            "Couldn't get engine versions for %s. Here's why: %s: %s",
            engine, err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return versions

def get_orderable_instances(self, db_engine, db_engine_version):
    """
    Gets DB instance options that can be used to create DB instances that are
    compatible with a set of specifications.

    :param db_engine: The database engine that must be supported by the DB
        instance.
    :param db_engine_version: The engine version that must be supported by the
        DB instance.
    :return: The list of DB instance options that can be used to create a
        compatible DB instance.
    """
    try:
        inst_opts = []
        paginator =
self.rds_client.get_paginator('describe_orderable_db_instance_options')
        for page in paginator.paginate(Engine=db_engine,
        EngineVersion=db_engine_version):
            inst_opts += page['OrderableDBInstanceOptions']
    except ClientError as err:
        logger.error(
            "Couldn't get orderable DB instances. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:

```

```
        return inst_opts

    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id)
            db_inst = response['DBInstances'][0]
        except ClientError as err:
            if err.response['Error']['Code'] == 'DBInstanceNotFound':
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response['Error']['Code'],
                    err.response['Error']['Message'])
                raise
        else:
            return db_inst

    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id, SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True)
            db_inst = response['DBInstance']
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
        else:
            return db_inst
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateDBCluster](#)
  - [CreateDBClusterParameterGroup](#)
  - [CreateDBClusterSnapshot](#)
  - [CreateDBInstance](#)
  - [DeleteDBCluster](#)
  - [DeleteDBClusterParameterGroup](#)
  - [DeleteDBInstance](#)
  - [DescribeDBClusterParameterGroups](#)
  - [DescribeDBClusterParameters](#)
  - [DescribeDBClusterSnapshots](#)
  - [DescribeDBClusters](#)
  - [DescribeDBEngineVersions](#)

- [DescribeDBInstances](#)
- [DescribeOrderableDBInstanceOptions](#)
- [ModifyDBClusterParameterGroup](#)

## Cross-service examples for Aurora using AWS SDKs

The following code examples show how to use Amazon Aurora with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Create an Aurora Serverless work item tracker \(p. 126\)](#)

## Create an Aurora Serverless work item tracker

The following code examples show how to create a web application that tracks work items in an Amazon Aurora Serverless database and uses Amazon Simple Email Service (Amazon SES) to send reports.

.NET

### AWS SDK for .NET

Shows how to use the AWS SDK for .NET to create a web application that tracks work items in an Amazon Aurora database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with a RESTful .NET backend.

- Integrate a React web application with AWS services.
- List, add, update, and delete items in an Aurora table.
- Send an email report of filtered work items using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

C++

### SDK for C++

Shows how to create a web application that tracks and reports on work items stored in an Amazon Aurora Serverless database.

For complete source code and instructions on how to set up a C++ REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

### Services used in this example

- Aurora

- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Java

### SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

For complete source code and instructions on how to set up and run an example that uses the JDBC API, see the full example on [GitHub](#).

#### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

JavaScript

### SDK for JavaScript V3

Shows how to use the AWS SDK for JavaScript (v3) to create a web application that tracks work items in an Amazon Aurora database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with an Express Node.js backend.

- Integrate a React.js web application with AWS services.
- List, add, and update items in an Aurora table.
- Send an email report of filtered work items by using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

#### **Services used in this example**

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

PHP

#### **SDK for PHP**

Shows how to use the AWS SDK for PHP to create a web application that tracks work items in an Amazon RDS database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with a RESTful PHP backend.

- Integrate a React.js web application with AWS services.
- List, add, update, and delete items in an Amazon RDS table.
- Send an email report of filtered work items using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Python

#### **SDK for Python (Boto3)**

Shows how to use the AWS SDK for Python (Boto3) to create a REST service that tracks work items in an Amazon Aurora Serverless database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses the Flask web framework to handle HTTP routing and integrates with a React webpage to present a fully functional web application.

- Build a Flask REST service that integrates with AWS services.
- Read, write, and update work items that are stored in an Aurora Serverless database.
- Create an AWS Secrets Manager secret that contains database credentials and use it to authenticate calls to the database.
- Use Amazon SES to send email reports of work items.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Aurora

- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

## Code examples for AWS Batch using AWS SDKs

The following code examples show how to use AWS Batch with an AWS software development kit (SDK).

### Code examples

- [Actions for AWS Batch using AWS SDKs \(p. 129\)](#)
  - [Describe one or more AWS Batch compute environments using an AWS SDK \(p. 129\)](#)

## Actions for AWS Batch using AWS SDKs

The following code examples show how to use AWS Batch with AWS SDKs. Each example calls an individual service function.

### Examples

- [Describe one or more AWS Batch compute environments using an AWS SDK \(p. 129\)](#)

## Describe one or more AWS Batch compute environments using an AWS SDK

The following code example shows how to describe one or more AWS Batch compute environments.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_envs(client: &Client) -> Result<(), Error> {
    let rsp = client.describe_compute_environments().send().await?;

    let compute_envs = rsp.compute_environments().unwrap_or_default();
    println!("Found {} compute environments:", compute_envs.len());
    for env in compute_envs {
        let arn = env.compute_environment_arn().unwrap_or_default();
        let name = env.compute_environment_name().unwrap_or_default();

        println!("  Name : {}", name);
        println!("  ARN:   {}", arn);
        println!("");
    }
    Ok(())
}
```

- For API details, see [DescribeComputeEnvironments](#) in *AWS SDK for Rust API reference*.

## Code examples for AWS CloudFormation using AWS SDKs

The following code examples show how to use AWS CloudFormation with an AWS software development kit (SDK).

### Code examples

- [Cross-service examples for AWS CloudFormation using AWS SDKs \(p. 130\)](#)
  - [Create an API Gateway REST API to track COVID-19 data \(p. 130\)](#)

## Cross-service examples for AWS CloudFormation using AWS SDKs

The following code examples show how to use AWS CloudFormation with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Create an API Gateway REST API to track COVID-19 data \(p. 130\)](#)

## Create an API Gateway REST API to track COVID-19 data

The following code example shows how to create a REST API that simulates a system to track daily cases of COVID-19 in the United States, using fictional data.

### Python

#### SDK for Python (Boto3)

Shows how to use AWS Chalice with the AWS SDK for Python (Boto3) to create a serverless REST API that uses Amazon API Gateway, AWS Lambda, and Amazon DynamoDB. The REST API simulates a system that tracks daily cases of COVID-19 in the United States, using fictional data. Learn how to:

- Use AWS Chalice to define routes in Lambda functions that are called to handle REST requests that come through API Gateway.
- Use Lambda functions to retrieve and store data in a DynamoDB table to serve REST requests.
- Define table structure and security role resources in an AWS CloudFormation template.
- Use AWS Chalice and CloudFormation to package and deploy all necessary resources.
- Use CloudFormation to clean up all created resources.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- API Gateway
- AWS CloudFormation
- DynamoDB
- Lambda

# Code examples for CloudFront using AWS SDKs

The following code examples show how to use Amazon CloudFront with an AWS software development kit (SDK).

## Code examples

- [Actions for CloudFront using AWS SDKs \(p. 131\)](#)
  - [Create a CloudFront function using an AWS SDK \(p. 131\)](#)
  - [Get CloudFront distribution configuration using an AWS SDK \(p. 132\)](#)
  - [List CloudFront distributions using an AWS SDK \(p. 132\)](#)
  - [Update a CloudFront distribution using an AWS SDK \(p. 133\)](#)

## Actions for CloudFront using AWS SDKs

The following code examples show how to use Amazon CloudFront with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create a CloudFront function using an AWS SDK \(p. 131\)](#)
- [Get CloudFront distribution configuration using an AWS SDK \(p. 132\)](#)
- [List CloudFront distributions using an AWS SDK \(p. 132\)](#)
- [Update a CloudFront distribution using an AWS SDK \(p. 133\)](#)

## Create a CloudFront function using an AWS SDK

The following code example shows how to create an Amazon CloudFront function.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createNewFunction(CloudFrontClient cloudFrontClient,
String functionName, String filePath) {
    try {
        InputStream is = new FileInputStream(filePath);
        SdkBytes functionCode = SdkBytes.fromInputStream(is);

        FunctionConfig config = FunctionConfig.builder()
            .comment("Created by using the CloudFront Java API")
            .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .name(functionName)
            .functionCode(functionCode)
            .functionConfig(config)
            .build();
    }
}
```

```
        CreateFunctionResponse response =
    cloudFrontClient.createFunction(functionRequest);
    return response.functionSummary().functionMetadata().functionARN();

} catch (CloudFrontException | FileNotFoundException e){
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [CreateFunction](#) in *AWS SDK for Java 2.x API Reference*.

## Get CloudFront distribution configuration using an AWS SDK

The following code example shows how to get Amazon CloudFront distribution configuration.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""
    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client

    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: ")

        distribution_config_response =
        self.cloudfront_client.get_distribution_config(
            Id=distribution_id)
        distribution_config = distribution_config_response['DistributionConfig']
        distribution_etag = distribution_config_response['ETag']

        distribution_config['Comment'] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: ")
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config, Id=distribution_id,
            IfMatch=distribution_etag)
        print("Done!")
```

- For API details, see [GetDistributionConfig](#) in *AWS SDK for Python (Boto3) API Reference*.

## List CloudFront distributions using an AWS SDK

The following code example shows how to list Amazon CloudFront distributions.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""
    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client

    def list_distributions(self):
        print("CloudFront distributions:\n")
        distributions = self.cloudfront_client.list_distributions()
        if distributions['DistributionList']['Quantity'] > 0:
            for distribution in distributions['DistributionList']['Items']:
                print(f"Domain: {distribution['DomainName']}")
                print(f"Distribution Id: {distribution['Id']}")
                print(f"Certificate Source: "
                      f"{distribution['ViewerCertificate']['CertificateSource']}")
                if distribution['ViewerCertificate']['CertificateSource'] == "acm":
                    print(f"Certificate: {distribution['ViewerCertificate']}"
                          ['Certificate'])
                else:
                    print("")
        else:
            print("No CloudFront distributions detected.")
```

- For API details, see [ListDistributions](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update a CloudFront distribution using an AWS SDK

The following code examples show how to update an Amazon CloudFront distribution.

### Java

#### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void modDistribution(CloudFrontClient cloudFrontClient, String idVal) {
    try {
        // Get the Distribution to modify.
        GetDistributionRequest disRequest = GetDistributionRequest.builder()
            .id(idVal)
            .build();

        GetDistributionResponse response =
            cloudFrontClient.getDistribution(disRequest);
        Distribution disObject = response.distribution();
        DistributionConfig config = disObject.distributionConfig();
```

```

        // Create a new DistributionConfig object and add new values to
comment and aliases
        DistributionConfig config1 = DistributionConfig.builder()
            .aliases(config.aliases()) // You can pass in new values here
            .comment("New Comment")
            .cacheBehaviors(config.cacheBehaviors())
            .priceClass(config.priceClass())
            .defaultCacheBehavior(config.defaultCacheBehavior())
            .enabled(config.enabled())
            .callerReference(config.callerReference())
            .logging(config.logging())
            .originGroups(config.originGroups())
            .origins(config.origins())
            .restrictions(config.restrictions())
            .defaultRootObject(config.defaultRootObject())
            .webACLId(config.webACLId())
            .httpVersion(config.httpVersion())
            .viewerCertificate(config.viewerCertificate())
            .customErrorResponses(config.customErrorResponses())
            .build();

        UpdateDistributionRequest updateDistributionRequest =
UpdateDistributionRequest.builder()
            .distributionConfig(config1)
            .id(disObject.id())
            .ifMatch(response.eTag())
            .build();

        cloudFrontClient.updateDistribution(updateDistributionRequest);

    } catch (CloudFrontException e){
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- For API details, see [UpdateDistribution](#) in *AWS SDK for Java 2.x API Reference*.

# Python

## SDK for Python (Boto3)

## Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudFrontWrapper:  
    """Encapsulates Amazon CloudFront operations."""  
    def __init__(self, cloudfront_client):  
        """  
        :param cloudfront_client: A Boto3 CloudFront client  
        """  
        self.cloudfront_client = cloudfront_client  
  
    def update_distribution(self):  
        distribution_id = input(  
            "This script updates the comment for a CloudFront distribution.\n"  
            "Enter a CloudFront distribution ID: ")  
  
        distribution_config_response =  
        self.cloudfront_client.get_distribution_config(  
            Id=distribution_id)
```

```
distribution_config = distribution_config_response['DistributionConfig']
distribution_etag = distribution_config_response['ETag']

distribution_config['Comment'] = input(
    f"\nThe current comment for distribution {distribution_id} is "
    f"'{distribution_config['Comment']}'.\n"
    f"Enter a new comment: ")
self.cloudfront_client.update_distribution(
    DistributionConfig=distribution_config, Id=distribution_id,
    IfMatch=distribution_etag)
print("Done!")
```

- For API details, see [UpdateDistribution](#) in *AWS SDK for Python (Boto3) API Reference*.

## Code examples for CloudWatch using AWS SDKs

The following code examples show how to use Amazon CloudWatch with an AWS software development kit (SDK).

### Code examples

- [Actions for CloudWatch using AWS SDKs \(p. 135\)](#)
  - Create a CloudWatch alarm that watches a metric using an AWS SDK (p. 136)
  - Delete CloudWatch alarms using an AWS SDK (p. 141)
  - Describe a CloudWatch alarm's history using an AWS SDK (p. 147)
  - Describe CloudWatch alarms using an AWS SDK (p. 148)
  - Describe CloudWatch alarms for a metric using an AWS SDK (p. 148)
  - Disable CloudWatch alarm actions using an AWS SDK (p. 153)
  - Enable CloudWatch alarm actions using an AWS SDK (p. 158)
  - Get the details of a CloudWatch dashboard (p. 164)
  - Get CloudWatch metric statistics using an AWS SDK (p. 165)
  - List CloudWatch dashboards (p. 166)
  - List CloudWatch metrics using an AWS SDK (p. 167)
  - Put a set of data into a CloudWatch metric using an AWS SDK (p. 174)
  - Put data into a CloudWatch metric using an AWS SDK (p. 174)
- [Scenarios for CloudWatch using AWS SDKs \(p. 178\)](#)
  - Get started with CloudWatch alarms using an AWS SDK (p. 179)
  - Manage CloudWatch metrics and alarms using an AWS SDK (p. 180)

## Actions for CloudWatch using AWS SDKs

The following code examples show how to use Amazon CloudWatch with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create a CloudWatch alarm that watches a metric using an AWS SDK \(p. 136\)](#)
- [Delete CloudWatch alarms using an AWS SDK \(p. 141\)](#)
- [Describe a CloudWatch alarm's history using an AWS SDK \(p. 147\)](#)
- [Describe CloudWatch alarms using an AWS SDK \(p. 148\)](#)
- [Describe CloudWatch alarms for a metric using an AWS SDK \(p. 148\)](#)

- [Disable CloudWatch alarm actions using an AWS SDK \(p. 153\)](#)
- [Enable CloudWatch alarm actions using an AWS SDK \(p. 158\)](#)
- [Get the details of a CloudWatch dashboard \(p. 164\)](#)
- [Get CloudWatch metric statistics using an AWS SDK \(p. 165\)](#)
- [List CloudWatch dashboards \(p. 166\)](#)
- [List CloudWatch metrics using an AWS SDK \(p. 167\)](#)
- [Put a set of data into a CloudWatch metric using an AWS SDK \(p. 174\)](#)
- [Put data into a CloudWatch metric using an AWS SDK \(p. 174\)](#)

## Create a CloudWatch alarm that watches a metric using an AWS SDK

The following code examples show how to create an Amazon CloudWatch alarm that watches a metric.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Create the alarm to watch the metric.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::PutMetricAlarmRequest request;
request.SetAlarmName(alarm_name);
request.SetComparisonOperator(
    Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
request.SetEvaluationPeriods(1);
request.SetMetricName("CPUUtilization");
request.SetNamespace("AWS/EC2");
request.SetPeriod(60);
request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
request.SetThreshold(70.0);
request.SetActionsEnabled(false);
request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("InstanceId");
dimension.SetValue(instanceId);

request.AddDimensions(dimension);

auto outcome = cw.PutMetricAlarm(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch alarm:" <<
        outcome.GetError().GetMessage() << std::endl;
```

```
        }
    else
    {
        std::cout << "Successfully created CloudWatch alarm " << alarm_name
        << std::endl;
    }
}
```

- For API details, see [PutMetricAlarm](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void putMetricAlarm(CloudWatchClient cw, String alarmName, String
instanceId) {

    try {
        Dimension dimension = Dimension.builder()
            .name("InstanceId")
            .value(instanceId).build();

        PutMetricAlarmRequest request = PutMetricAlarmRequest.builder()
            .alarmName(alarmName)
            .comparisonOperator(ComparisonOperator.GREATER_THAN_THRESHOLD)
            .evaluationPeriods(1)
            .metricName("CPUUtilization")
            .namespace("AWS/EC2")
            .period(60)
            .statistic(Statistic.AVERAGE)
            .threshold(70.0)
            .actionsEnabled(false)
            .alarmDescription("Alarm when server CPU utilization exceeds 70%")
            .unit(StandardUnit.SECONDS)
            .dimensions(dimension)
            .build();

        cw.putMetricAlarm(request);
        System.out.printf("Successfully created alarm with name %s",
alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutMetricAlarm](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { PutMetricAlarmCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanOrEqualToThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: false,
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

export const run = async () => {
  try {
    const data = await cwClient.send(new PutMetricAlarmCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutMetricAlarm](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
```

```
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
    AlarmName: 'Web_Server_CPU_Utilization',
    ComparisonOperator: 'GreaterThanThreshold',
    EvaluationPeriods: 1,
    MetricName: 'CPUUtilization',
    Namespace: 'AWS/EC2',
    Period: 60,
    Statistic: 'Average',
    Threshold: 70.0,
    ActionsEnabled: false,
    AlarmDescription: 'Alarm when server CPU exceeds 70%',
    Dimensions: [
        {
            Name: 'InstanceId',
            Value: 'INSTANCE_ID'
        },
    ],
    Unit: 'Percent'
};

cw.putMetricAlarm(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutMetricAlarm](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun putMetricAlarm(alarmNameVal: String, instanceIdVal: String) {

    val dimension0b = Dimension {
        name = "InstanceId"
        value = instanceIdVal
    }

    val request = PutMetricAlarmRequest {
        alarmName = alarmNameVal
        comparisonOperator = ComparisonOperator.GreaterThanThreshold
        evaluationPeriods = 1
    }
}
```

```
        metricName = "CPUUtilization"
        namespace = "AWS/EC2"
        period = 60
        statistic = Statistic.fromValue("Average")
        threshold = 70.0
        actionsEnabled = false
        alarmDescription = "An Alarm created by the Kotlin SDK when server CPU
utilization exceeds 70%"
        unit = StandardUnit.fromValue("Seconds")
        dimensions = listOf(dimension0b)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
}
```

- For API details, see [PutMetricAlarm](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""
    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource

    def create_metric_alarm(
            self, metric_namespace, metric_name, alarm_name, stat_type, period,
            eval_periods, threshold, comparison_op):
        """
        Creates an alarm that watches a metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
        :param period: The period in which metric data are grouped to calculate
                       statistics.
        :param eval_periods: The number of periods that the metric must be over the
                           alarm threshold before the alarm is set into an
                           alarmed
                           state.
        :param threshold: The threshold value to compare against the metric
                          statistic.
        :param comparison_op: The comparison operation used to compare the
                              threshold
                              against the metric.
        :return: The newly created alarm.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
            alarm = metric.put_alarm(
```

```
AlarmName=alarm_name,
Statistic=stat_type,
Period=period,
EvaluationPeriods=eval_periods,
Threshold=threshold,
ComparisonOperator=comparison_op)
logger.info(
    "Added alarm %s to track metric %s.%s.", alarm_name,
metric_namespace,
    metric_name)
except ClientError:
    logger.exception(
        "Couldn't add alarm %s to metric %s.%s", alarm_name,
metric_namespace,
    metric_name)
    raise
else:
    return alarm
```

- For API details, see [PutMetricAlarm](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  lo_cwt->putmetricalarm(
    iv_alarmname          = iv_alarm_name
    iv_comparisonoperator = iv_comparison_operator
    iv_evaluationperiods = iv_evaluation_periods
    iv_metricname         = iv_metric_name
    iv_namespace          = iv_namespace
    iv_statistic          = iv_statistic
    iv_threshold          = iv_threshold
    iv_actionsenabled    = iv_actions_enabled
    iv_alarmdescription  = iv_alarm_description
    iv_unit               = iv_unit
    iv_period              = iv_period
    it_dimensions          = it_dimensions
  ).
  MESSAGE 'Alarm created' TYPE 'I'.
CATCH /aws1/cx_cwtlimitexceededfault.
  MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
ENDTRY.
```

- For API details, see [PutMetricAlarm](#) in *AWS SDK for SAP ABAP API reference*.

## Delete CloudWatch alarms using an AWS SDK

The following code examples show how to delete Amazon CloudWatch alarms.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// This example shows how to delete Amazon CloudWatch alarms. The example
/// was created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DeleteAlarms
{
    public static async Task Main()
    {
        IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();

        var alarmNames = CreateAlarmNameList();
        await DeleteAlarmsAsyncExample(cwClient, alarmNames);
    }

    /// <summary>
    /// Delete the alarms whose names are listed in the alarmNames parameter.
    /// </summary>
    /// <param name="client">The initialized Amazon CloudWatch client.</param>
    /// <param name="alarmNames">A list of names for the alarms to be
    /// deleted.</param>
    public static async Task DeleteAlarmsAsyncExample(IAmazonCloudWatch client,
List<string> alarmNames)
    {
        var request = new DeleteAlarmsRequest
        {
            AlarmNames = alarmNames,
        };

        try
        {
            var response = await client.DeleteAlarmsAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("Alarms successfully deleted:");
                alarmNames
                    .ForEach(name => Console.WriteLine($"'{name}'"));
            }
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
    }

    /// <summary>
    /// Defines and returns the list of alarm names to delete.
    /// </summary>
    /// <returns>A list of alarm names.</returns>
    public static List<string> CreateAlarmNameList()
```

```
{  
    // The list of alarm names passed to DeleteAlarmsAsync  
    // can contain up to 100 alarm names.  
    var theList = new List<string>  
    {  
        "ALARM_NAME_1",  
        "ALARM_NAME_2",  
    };  
  
    return theList;  
}  
}
```

- For API details, see [DeleteAlarms](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>  
#include <aws/monitoring/CloudWatchClient.h>  
#include <aws/monitoring/model/DeleteAlarmsRequest.h>  
#include <iostream>
```

Delete the alarm.

```
Aws::CloudWatch::CloudWatchClient cw;  
Aws::CloudWatch::Model::DeleteAlarmsRequest request;  
request.AddAlarmNames(alarm_name);  
  
auto outcome = cw.DeleteAlarms(request);  
if (!outcome.IsSuccess())  
{  
    std::cout << "Failed to delete CloudWatch alarm:" <<  
        outcome.GetError().GetMessage() << std::endl;  
}  
else  
{  
    std::cout << "Successfully deleted CloudWatch alarm " << alarm_name  
        << std::endl;  
}
```

- For API details, see [DeleteAlarms](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteCWAAlarm(CloudWatchClient cw, String alarmName) {  
  
    try {  
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()  
            .alarmNames(alarmName)  
            .build();  
  
        cw.deleteAlarms(request);  
        System.out.printf("Successfully deleted alarm %s", alarmName);  
  
    } catch (CloudWatchException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteAlarms](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create an Amazon CloudWatch service client object.  
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js  
import { DeleteAlarmsCommand } from "@aws-sdk/client-cloudwatch";  
import { cwClient } from "./libs/cloudWatchClient.js";  
  
// Set the parameters  
export const params = { AlarmNames: "ALARM_NAME" }; // e.g.,  
// "Web_Server_CPU_Utilization"  
  
export const run = async () => {  
    try {  
        const data = await cwClient.send(new DeleteAlarmsCommand(params));  
        console.log("Success, alarm deleted; requestID:", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
  
// Uncomment this line to run execution within this file.  
// run();
```

- 
- For more information, see [AWS SDK for JavaScript Developer Guide](#).
  - For API details, see [DeleteAlarms](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Import the SDK and client modules and call the API.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
  AlarmNames: ['Web_Server_CPU_Utilization']
};

cw.deleteAlarms(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 
- For more information, see [AWS SDK for JavaScript Developer Guide](#).
  - For API details, see [DeleteAlarms](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteCWAAlarm(alarmNameVal: String) {

    val request = DeleteAlarmsRequest {
        alarmNames = listOf(alarmNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- For API details, see [DeleteAlarms](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:  
    """Encapsulates Amazon CloudWatch functions."""  
    def __init__(self, cloudwatch_resource):  
        """  
        :param cloudwatch_resource: A Boto3 CloudWatch resource.  
        """  
        self.cloudwatch_resource = cloudwatch_resource  
  
    def delete_metric_alarms(self, metric_namespace, metric_name):  
        """  
        Deletes all of the alarms that are currently watching the specified metric.  
        :param metric_namespace: The namespace of the metric.  
        :param metric_name: The name of the metric.  
        """  
        try:  
            metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)  
            metric.alarms.delete()  
            logger.info(  
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name)  
        except ClientError:  
            logger.exception(  
                "Couldn't delete alarms for metric %s.%s.", metric_namespace,  
                metric_name)  
            raise
```

- For API details, see [DeleteAlarms](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_cwt->deletealarms(  
        it_alarmnames = it_alarm_names  
    ).  
    MESSAGE 'Alarms deleted' TYPE 'I'.  
CATCH /aws1/cx_cwtresourcenotfound .  
    MESSAGE 'Resource being access is not found.' TYPE 'E'.
```

ENDTRY.

- For API details, see [DeleteAlarms](#) in [AWS SDK for SAP ABAP API reference](#).

## Describe a CloudWatch alarm's history using an AWS SDK

The following code example shows how to describe Amazon CloudWatch alarm history.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get a list of CloudWatch alarms, then retrieve the history for each alarm.

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

///<summary>
/// This example retrieves a list of Amazon CloudWatch alarms and, for
/// each one, displays its history. The example was created using the
/// AWS SDK for .NET 3.7 and .NET Core 5.0.
///</summary>
public class DescribeAlarmHistories
{
    ///<summary>
    /// Retrieves a list of alarms and then passes each name to the
    /// DescribeAlarmHistoriesAsync method to retrieve its history.
    ///</summary>
    public static async Task Main()
    {
        IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();
        var response = await cwClient.DescribeAlarmsAsync();

        foreach (var alarm in response.MetricAlarms)
        {
            await DescribeAlarmHistoriesAsync(cwClient, alarm.AlarmName);
        }
    }

    ///<summary>
    /// Retrieves the CloudWatch alarm history for the alarm name passed
    /// to the method.
    ///</summary>
    ///<param name="client">An initialized CloudWatch client object.</param>
    ///<param name="alarmName">The CloudWatch alarm for which to retrieve
    /// history information.</param>
    public static async Task DescribeAlarmHistoriesAsync(IAmazonCloudWatch
client, string alarmName)
    {
        var request = new DescribeAlarmHistoryRequest
        {
            AlarmName = alarmName,
            EndDateUtc = DateTime.Today,
            HistoryItemType = HistoryItemType.Action,
            MaxRecords = 1,
            StartDateUtc = DateTime.Today.Subtract(TimeSpan.FromDays(30)),
        }
    }
}
```

```
};

var response = new DescribeAlarmHistoryResponse();

do
{
    response = await client.DescribeAlarmHistoryAsync(request);

    foreach (var item in response.AlarmHistoryItems)
    {
        Console.WriteLine(item.AlarmName);
        Console.WriteLine(item.HistorySummary);
        Console.WriteLine();
    }

    request.NextToken = response.NextToken;
}
while (!string.IsNullOrEmpty(response.NextToken));
}
```

- For API details, see [DescribeAlarmHistory in AWS SDK for .NET API Reference](#).

## Describe CloudWatch alarms using an AWS SDK

The following code example shows how to describe Amazon CloudWatch alarms.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_cwt->describealarms(                               " oo_result is returned
for testing purpose "
        it_alarmnames = it_alarm_names
    ).
    MESSAGE 'Alarms retrieved' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DescribeAlarms in AWS SDK for SAP ABAP API reference](#).

## Describe CloudWatch alarms for a metric using an AWS SDK

The following code examples show how to describe Amazon CloudWatch alarms for a metric.

C++

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DescribeAlarmsRequest.h>
#include <aws/monitoring/model/DescribeAlarmsResult.h>
#include <iomanip>
#include <iostream>
```

Describe the alarms.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::DescribeAlarmsRequest request;
request.SetMaxRecords(1);

bool done = false;
bool header = false;
while (!done)
{
    auto outcome = cw.DescribeAlarms(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to describe CloudWatch alarms:" <<
            outcome.GetError().GetMessage() << std::endl;
        break;
    }

    if (!header)
    {
        std::cout << std::left <<
            std::setw(32) << "Name" <<
            std::setw(64) << "Arn" <<
            std::setw(64) << "Description" <<
            std::setw(20) << "LastUpdated" <<
            std::endl;
        header = true;
    }

    const auto &alarms = outcome.GetResult().GetMetricAlarms();
    for (const auto &alarm : alarms)
    {
        std::cout << std::left <<
            std::setw(32) << alarm.GetAlarmName() <<
            std::setw(64) << alarm.GetAlarmArn() <<
            std::setw(64) << alarm.GetAlarmDescription() <<
            std::setw(20) <<
            alarm.GetAlarmConfigurationUpdatedTimestamp().ToGmtString(
                SIMPLE_DATE_FORMAT_STR) <<
            std::endl;
    }

    const auto &next_token = outcome.GetResult().GetNextToken();
    request.SetNextToken(next_token);
    done = next_token.empty();
}
```

- For API details, see [DescribeAlarmsForMetric](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void desCWAlarms( CloudWatchClient cw) {  
    try {  
  
        boolean done = false;  
        String newToken = null;  
  
        while(!done) {  
            DescribeAlarmsResponse response;  
            if (newToken == null) {  
                DescribeAlarmsRequest request =  
                    DescribeAlarmsRequest.builder().build();  
                response = cw.describeAlarms(request);  
            } else {  
                DescribeAlarmsRequest request = DescribeAlarmsRequest.builder()  
                    .nextToken(newToken)  
                    .build();  
                response = cw.describeAlarms(request);  
            }  
  
            for(MetricAlarm alarm : response.metricAlarms()) {  
                System.out.printf("\n Retrieved alarm %s", alarm.alarmName());  
            }  
  
            if(response.nextToken() == null) {  
                done = true;  
            } else {  
                newToken = response.nextToken();  
            }  
        }  
  
    } catch (CloudWatchException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    System.out.printf("Done");  
}
```

- For API details, see [DescribeAlarmsForMetric](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { DescribeAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = { StateValue: "INSUFFICIENT_DATA" };

export const run = async () => {
    try {
        const data = await cwClient.send(new DescribeAlarmsCommand(params));
        console.log("Success", data);
        data.MetricAlarms.forEach(function (item) {
            console.log(item.AlarmName);
        });
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeAlarmsForMetric](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

cw.describeAlarms({StateValue: 'INSUFFICIENT_DATA'}, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        // List the names of all current alarms in the console
        data.MetricAlarms.forEach(function (item, index, array) {
            console.log(item.AlarmName);
        });
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeAlarmsForMetric](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun desCWAAlarms() {  
  
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->  
        val response = cwClient.describeAlarms(DescribeAlarmsRequest {})  
        response.metricAlarms?.forEach { alarm ->  
            println("Retrieved alarm ${alarm.alarmName}")  
        }  
    }  
}
```

- For API details, see [DescribeAlarmsForMetric](#) in [AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:  
    """Encapsulates Amazon CloudWatch functions."""  
    def __init__(self, cloudwatch_resource):  
        """  
        :param cloudwatch_resource: A Boto3 CloudWatch resource.  
        """  
        self.cloudwatch_resource = cloudwatch_resource  
  
    def get_metric_alarms(self, metric_namespace, metric_name):  
        """  
        Gets the alarms that are currently watching the specified metric.  
  
        :param metric_namespace: The namespace of the metric.  
        :param metric_name: The name of the metric.  
        :returns: An iterator that yields the alarms.  
        """  
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)  
        alarm_iter = metric.alarms.all()  
        logger.info("Got alarms for metric %s.%s.", metric_namespace, metric_name)  
        return alarm_iter
```

- For API details, see [DescribeAlarmsForMetric](#) in *AWS SDK for Python (Boto3) API Reference*.

## Disable CloudWatch alarm actions using an AWS SDK

The following code examples show how to disable Amazon CloudWatch alarm actions.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// This example shows how to disable the Amazon CloudWatch actions for
/// one or more CloudWatch alarms. The example was created using the
/// AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DisableAlarmActions
{
    public static async Task Main()
    {
        IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();
        var alarmNames = new List<string>
        {
            "ALARM_NAME",
            "ALARM_NAME_2",
        };

        var success = await DisableAlarmsActionsAsync(cwClient, alarmNames);

        if (success)
        {
            Console.WriteLine("Alarm action(s) successfully disabled.");
        }
        else
        {
            Console.WriteLine("Alarm action(s) were not disabled.")
        }
    }

    /// <summary>
    /// Disable the actions for the list of CloudWatch alarm names passed
    /// in the alarmNames parameter.
    /// </summary>
    /// <param name="client">An initialized CloudWatch client object.</param>
    /// <param name="alarmNames">The list of CloudWatch alarms to disable.</param>
    /// <returns>A Boolean value indicating the success of the call.</returns>
    public static async Task<bool> DisableAlarmsActionsAsync(
        IAmazonCloudWatch client,
        List<string> alarmNames)
    {
        var request = new DisableAlarmActionsRequest
        {
```

```
        AlarmNames = alarmNames,
    };

    var response = await client.DisableAlarmActionsAsync(request);

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [DisableAlarmActions](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DisableAlarmActionsRequest.h>
#include <iostream>
```

Disable the alarm actions.

```
Aws::CloudWatch::CloudWatchClient cw;

Aws::CloudWatch::Model::DisableAlarmActionsRequest
disableAlarmActionsRequest;
disableAlarmActionsRequest.AddAlarmNames(alarm_name);

auto disableAlarmActionsOutcome =
cw.DisableAlarmActions(disableAlarmActionsRequest);
if (!disableAlarmActionsOutcome.IsSuccess())
{
    std::cout << "Failed to disable actions for alarm " << alarm_name <<
    ":" << disableAlarmActionsOutcome.GetError().GetMessage() <<
    std::endl;
}
else
{
    std::cout << "Successfully disabled actions for alarm " <<
    alarm_name << std::endl;
}
```

- For API details, see [DisableAlarmActions](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void disableActions(CloudWatchClient cw, String alarmName) {  
  
    try {  
        DisableAlarmActionsRequest request =  
DisableAlarmActionsRequest.builder()  
            .alarmNames(alarmName)  
            .build();  
  
        cw.disableAlarmActions(request);  
        System.out.printf("Successfully disabled actions on alarm %s",  
alarmName);  
  
    } catch (CloudWatchException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DisableAlarmActions](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create an Amazon CloudWatch service client object.  
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js  
import { DisableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";  
import { cwClient } from "./libs/cloudWatchClient.js";  
  
// Set the parameters  
export const params = { AlarmNames: "ALARM_NAME" }; // e.g.,  
// "Web_Server_CPU_Utilization"  
  
export const run = async () => {  
    try {  
        const data = await cwClient.send(new DisableAlarmActionsCommand(params));  
        console.log("Success, alarm disabled:", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
// Uncomment this line to run execution within this file.
```

```
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DisableAlarmActions](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Import the SDK and client modules and call the API.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

cw.disableAlarmActions({AlarmNames: ['Web_Server_CPU_Utilization']}, function(err,
  data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DisableAlarmActions](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun disableActions(alarmName: String) {

    val request = DisableAlarmActionsRequest {
        alarmNames = listOf(alarmName)
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

- For API details, see [DisableAlarmActions](#) in [AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:  
    """Encapsulates Amazon CloudWatch functions."""  
    def __init__(self, cloudwatch_resource):  
        """  
        :param cloudwatch_resource: A Boto3 CloudWatch resource.  
        """  
        self.cloudwatch_resource = cloudwatch_resource  
  
    def enable_alarm_actions(self, alarm_name, enable):  
        """  
        Enables or disables actions on the specified alarm. Alarm actions can be  
        used to send notifications or automate responses when an alarm enters a  
        particular state.  
  
        :param alarm_name: The name of the alarm.  
        :param enable: When True, actions are enabled for the alarm. Otherwise,  
        they  
                disabled.  
        """  
        try:  
            alarm = self.cloudwatch_resource.Alarm(alarm_name)  
            if enable:  
                alarm.enable_actions()  
            else:  
                alarm.disable_actions()  
            logger.info(  
                "%s actions for alarm %s.", "Enabled" if enable else "Disabled",  
                alarm_name)  
        except ClientError:  
            logger.exception(  
                "Couldn't %s actions alarm %s.", "enable" if enable else "disable",  
                alarm_name)  
        raise
```

- For API details, see [DisableAlarmActions](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Disables actions on the specified alarm. "  
TRY.  
    lo_cwt->disablealarmactions(
```

```
        it_alarmnames = it_alarm_names
    ).
    MESSAGE 'Alarm actions disabled' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DisableAlarmActions](#) in *AWS SDK for SAP ABAP API reference*.

## Enable CloudWatch alarm actions using an AWS SDK

The following code examples show how to enable Amazon CloudWatch alarm actions.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// This example shows how to enable the Amazon CloudWatch actions for
/// one or more CloudWatch alarms. The example was created using the
/// AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class EnableAlarmActions
{
    public static async Task Main()
    {
        IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();
        var alarmNames = new List<string>
        {
            "ALARM_NAME",
            "ALARM_NAME_2",
        };

        var success = await EnableAlarmActionsAsync(cwClient, alarmNames);

        if (success)
        {
            Console.WriteLine("Alarm action(s) successfully enabled.");
        }
        else
        {
            Console.WriteLine("Alarm action(s) were not enabled.")
        }
    }

    /// <summary>
    /// Enable the actions for the list of CloudWatch alarm names passed
    /// in the alarmNames parameter.
    /// </summary>
```

```
    /// <param name="client">An initialized CloudWatch client object.</param>
    /// <param name="alarmNames">The list of CloudWatch alarms to enable.</param>
    public static async Task<bool> EnableAlarmActionsAsync(IAmazonCloudWatch
client, List<string> alarmNames)
{
    var request = new EnableAlarmActionsRequest
    {
        AlarmNames = alarmNames,
    };

    var response = await client.EnableAlarmActionsAsync(request);

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/EnableAlarmActionsRequest.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Enable the alarm actions.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::PutMetricAlarmRequest request;
request.SetAlarmName(alarm_name);
request.SetComparisonOperator(
    Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
request.SetEvaluationPeriods(1);
request.SetMetricName("CPUUtilization");
request.SetNamespace("AWS/EC2");
request.SetPeriod(60);
request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
request.SetThreshold(70.0);
request.SetActionsEnabled(false);
request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);
request.AddAlarmActions(actionArn);

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("InstanceId");
dimension.SetValue(instanceId);
request.AddDimensions(dimension);
```

```
auto outcome = cw.PutMetricAlarm(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch alarm:" <<
        outcome.GetError().GetMessage() << std::endl;
    return;
}

Aws::CloudWatch::Model::EnableAlarmActionsRequest enable_request;
enable_request.AddAlarmNames(alarm_name);

auto enable_outcome = cw.EnableAlarmActions(enable_request);
if (!enable_outcome.IsSuccess())
{
    std::cout << "Failed to enable alarm actions:" <<
        enable_outcome.GetError().GetMessage() << std::endl;
    return;
}

std::cout << "Successfully created alarm " << alarm_name <<
    " and enabled actions on it." << std::endl;
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void enableActions(CloudWatchClient cw, String alarm) {

    try {
        EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()
            .alarmNames(alarm)
            .build();

        cw.enableAlarmActions(request);
        System.out.printf("Successfully enabled actions on alarm %s", alarm);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {
  PutMetricAlarmCommand,
  EnableAlarmActionsCommand,
} from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = {
  AlarmName: "ALARM_NAME", //ALARM_NAME
  ComparisonOperator: "GreaterThanOrEqualToThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: true,
  AlarmActions: ["ACTION_ARN"], //e.g., "arn:aws:automate:us-east-1:ec2:stop"
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

export const run = async () => {
  try {
    const data = await cwClient.send(new PutMetricAlarmCommand(params));
    console.log("Alarm action added; RequestID:", data);
    return data;
  } catch (err) {
    console.log("Error", err);
    return data;
  }
} catch (err) {
  console.log("Error", err);
}
};

// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [EnableAlarmActions](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Import the SDK and client modules and call the API.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
    AlarmName: 'Web_Server_CPU_Utilization',
    ComparisonOperator: 'GreaterThanOrEqualToThreshold',
    EvaluationPeriods: 1,
    MetricName: 'CPUUtilization',
    Namespace: 'AWS/EC2',
    Period: 60,
    Statistic: 'Average',
    Threshold: 70.0,
    ActionsEnabled: true,
    AlarmActions: ['ACTION_ARN'],
    AlarmDescription: 'Alarm when server CPU exceeds 70%',
    Dimensions: [
        {
            Name: 'InstanceId',
            Value: 'INSTANCE_ID'
        },
    ],
    Unit: 'Percent'
};

cw.putMetricAlarm(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Alarm action added", data);
        var paramsEnableAlarmAction = {
            AlarmNames: [params.AlarmName]
        };
        cw.enableAlarmActions(paramsEnableAlarmAction, function(err, data) {
            if (err) {
                console.log("Error", err);
            } else {
                console.log("Alarm action enabled", data);
            }
        });
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [EnableAlarmActions](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun enableActions(alarm: String) {  
  
    val request = EnableAlarmActionsRequest {  
        alarmNames = listOf(alarm)  
    }  
  
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->  
        cwClient.enableAlarmActions(request)  
        println("Successfully enabled actions on alarm $alarm")  
    }  
}
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:  
    """Encapsulates Amazon CloudWatch functions."""  
    def __init__(self, cloudwatch_resource):  
        """  
        :param cloudwatch_resource: A Boto3 CloudWatch resource.  
        """  
        self.cloudwatch_resource = cloudwatch_resource  
  
    def enable_alarm_actions(self, alarm_name, enable):  
        """  
        Enables or disables actions on the specified alarm. Alarm actions can be  
        used to send notifications or automate responses when an alarm enters a  
        particular state.  
  
        :param alarm_name: The name of the alarm.  
        :param enable: When True, actions are enabled for the alarm. Otherwise,  
        they  
                    disabled.  
        """  
        try:  
            alarm = self.cloudwatch_resource.Alarm(alarm_name)  
            if enable:  
                alarm.enable_actions()  
            else:  
                alarm.disable_actions()  
            logger.info(  
                "%s actions for alarm %s.", "Enabled" if enable else "Disabled",
```

```
        alarm_name)
    except ClientError:
        logger.exception(
            "Couldn't %s actions alarm %s.", "enable" if enable else "disable",
            alarm_name)
        raise
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Enable actions on the specified alarm."
TRY.
    lo_cwt->enablealarmactions(
        it_alarmnames = it_alarm_names
    ).
    MESSAGE 'Alarm actions enabled' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for SAP ABAP API reference*.

## Get the details of a CloudWatch dashboard

The following code example shows how to get Amazon CloudWatch dashboard details.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// This example shows how to retrieve the details of an Amazon CloudWatch
/// dashboard. The return value from the call to GetDashboard is a json
/// object representing the widgets in the dashboard. The example was
```

```
/// created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class GetDashboard
{
    public static async Task Main()
    {
        IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();
        string dashboardName = "CloudWatch-Default";

        var body = await GetDashboardAsync(cwClient, dashboardName);

        Console.WriteLine(body);
    }

    /// <summary>
    /// Get the json that represents the dashboard.
    /// </summary>
    /// <param name="client">An initialized CloudWatch client.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>The string containing the json value describing the
    /// contents and layout of the CloudWatch dashboard.</returns>
    public static async Task<string> GetDashboardAsync(IAmazonCloudWatch
client, string dashboardName)
    {

        var request = new GetDashboardRequest
        {
            DashboardName = dashboardName,
        };

        var response = await client.GetDashboardAsync(request);

        return response.DashboardBody;
    }
}
```

- For API details, see [GetDashboard](#) in *AWS SDK for .NET API Reference*.

## Get CloudWatch metric statistics using an AWS SDK

The following code example shows how to get Amazon CloudWatch metric statistics.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""
    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource

    def get_metric_statistics(self, namespace, name, start, end, period,
                           stat_types):
```

```
"""
    Gets statistics for a metric within a specified time span. Metrics are
grouped
        into the specified period.

    :param namespace: The namespace of the metric.
    :param name: The name of the metric.
    :param start: The UTC start time of the time span to retrieve.
    :param end: The UTC end time of the time span to retrieve.
    :param period: The period, in seconds, in which to group metrics. The
period
                    must match the granularity of the metric, which depends on
the metric's age. For example, metrics that are older than
three hours have a one-minute granularity, so the period
must
                    be at least 60 and must be a multiple of 60.
    :param stat_types: The type of statistics to retrieve, such as average
value
                    or maximum value.
    :return: The retrieved statistics for the metric.
"""
try:
    metric = self.cloudwatch_resource.Metric(namespace, name)
    stats = metric.get_statistics(
        StartTime=start, EndTime=end, Period=period, Statistics=stat_types)
    logger.info(
        "Got %s statistics for %s.", len(stats['Datapoints']),
        stats['Label'])
except ClientError:
    logger.exception("Couldn't get statistics for %s.%s.", namespace, name)
    raise
else:
    return stats
```

- For API details, see [GetMetricStatistics](#) in *AWS SDK for Python (Boto3) API Reference*.

## List CloudWatch dashboards

The following code example shows how to list Amazon CloudWatch dashboards.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// Shows how to retrieve a list of Amazon CloudWatch dashboards. This
/// example was written using AWSSDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class ListDashboards
{
```

```
public static async Task Main()
{
    IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();
    var dashboards = await ListDashboardsAsync(cwClient);

    DisplayDashboardList(dashboards);
}

/// <summary>
/// Get the list of available dashboards.
/// </summary>
/// <param name="client">The initialized CloudWatch client used to
/// retrieve a list of defined dashboards.</param>
/// <returns>A list of DashboardEntry objects.</returns>
public static async Task<List<DashboardEntry>>
ListDashboardsAsync(IAmazonCloudWatch client)
{
    var response = await client.ListDashboardsAsync(new
ListDashboardsRequest());
    return response.DashboardEntries;
}

/// <summary>
/// Displays the name of each CloudWatch Dashboard in the list passed
/// to the method.
/// </summary>
/// <param name="dashboards">A list of DashboardEntry objects.</param>
public static void DisplayDashboardList(List<DashboardEntry> dashboards)
{
    if (dashboards.Count > 0)
    {
        Console.WriteLine("The following dashboards are defined:");
        foreach (var dashboard in dashboards)
        {
            Console.WriteLine($"Name: {dashboard.DashboardName} Last
modified: {dashboard.LastModified}");
        }
    }
    else
    {
        Console.WriteLine("No dashboards found.");
    }
}
```

- For API details, see [ListDashboards](#) in *AWS SDK for .NET API Reference*.

## List CloudWatch metrics using an AWS SDK

The following code examples show how to list Amazon CloudWatch metrics.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
```

```
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// This example demonstrates how to list metrics for Amazon CloudWatch.
/// The example was created using the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
/// </summary>
public class ListMetrics
{
    public static async Task Main()
    {
        IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();

        var filter = new DimensionFilter
        {
            Name = "InstanceType",
            Value = "t1.micro",
        };
        string metricName = "CPUUtilization";
        string namespaceName = "AWS/EC2";

        await ListMetricsAsync(cwClient, filter, metricName, namespaceName);
    }

    /// <summary>
    /// Retrieve CloudWatch metrics using the supplied filter, metrics name,
    /// and namespace.
    /// </summary>
    /// <param name="client">An initialized CloudWatch client.</param>
    /// <param name="filter">The filter to apply in retrieving metrics.</param>
    /// <param name="metricName">The metric name for which to retrieve
    /// information.</param>
    /// <param name="nameSpaceName">The name of the namespace from which
    /// to retrieve metric information.</param>
    public static async Task ListMetricsAsync(
        IAmazonCloudWatch client,
        DimensionFilter filter,
        string metricName,
        string nameSpaceName)
    {
        var request = new ListMetricsRequest
        {
            Dimensions = new List<DimensionFilter>() { filter },
            MetricName = metricName,
            Namespace = nameSpaceName,
        };

        var response = new ListMetricsResponse();
        do
        {
            response = await client.ListMetricsAsync(request);

            if (response.Metrics.Count > 0)
            {
                foreach (var metric in response.Metrics)
                {
                    Console.WriteLine(metric.MetricName +
                        " (" + metric.Namespace + ")");

                    foreach (var dimension in metric.Dimensions)
                    {
                        Console.WriteLine(" " + dimension.Name + ":" +
                            dimension.Value);
                    }
                }
            }
        } while (response.NextToken != null);
    }
}
```

```
        }
    }
}
else
{
    Console.WriteLine("No metrics found.");
}

request.NextToken = response.NextToken;
}
while (!string.IsNullOrEmpty(response.NextToken));
}
```

- For API details, see [ListMetrics](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/ListMetricsRequest.h>
#include <aws/monitoring/model/ListMetricsResult.h>
#include <iomanip>
#include <iostream>
```

List the metrics.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::ListMetricsRequest request;

if (argc > 1)
{
    request.SetMetricName(argv[1]);
}

if (argc > 2)
{
    request.SetNamespace(argv[2]);
}

bool done = false;
bool header = false;
while (!done)
{
    auto outcome = cw.ListMetrics(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to list CloudWatch metrics:" <<
            outcome.GetError().GetMessage() << std::endl;
        break;
}
```

```
if (!header)
{
    std::cout << std::left << std::setw(48) << "MetricName" <<
        std::setw(32) << "Namespace" << "DimensionNameValuePairs" <<
        std::endl;
    header = true;
}

const auto &metrics = outcome.GetResult().GetMetrics();
for (const auto &metric : metrics)
{
    std::cout << std::left << std::setw(48) <<
        metric.GetMetricName() << std::setw(32) <<
        metric.GetNamespace();
    const auto &dimensions = metric.GetDimensions();
    for (auto iter = dimensions.cbegin();
        iter != dimensions.cend(); ++iter)
    {
        const auto &dimkv = *iter;
        std::cout << dimkv.GetName() << " = " << dimkv.GetValue();
        if (iter + 1 != dimensions.cend())
        {
            std::cout << ", ";
        }
    }
    std::cout << std::endl;
}

const auto &next_token = outcome.GetResult().GetNextToken();
request.SetNextToken(next_token);
done = next_token.empty();
}
```

- For API details, see [ListMetrics](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listMets( CloudWatchClient cw, String namespace) {

    boolean done = false;
    String nextToken = null;

    try {
        while(!done) {

            ListMetricsResponse response;
            if (nextToken == null) {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

                response = cw.listMetrics(request);
            } else {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
```

```
        .nextToken(nextToken)
        .build();

        response = cw.listMetrics(request);
    }

    for (Metric metric : response.metrics()) {
        System.out.printf("Retrieved metric %s", metric.metricName());
        System.out.println();
    }

    if(response.nextToken() == null) {
        done = true;
    } else {
        nextToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [ListMetrics](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { ListMetricsCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = {
  Dimensions: [
    {
      Name: "LogGroupName" /* required */,
    },
  ],
  MetricName: "IncomingLogEvents",
  Namespace: "AWS/Logs",
```

```
};

export const run = async () => {
  try {
    const data = await cwClient.send(new ListMetricsCommand(params));
    console.log("Success. Metrics:", JSON.stringify(data.Metrics));
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};

// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListMetrics](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
  Dimensions: [
    {
      Name: 'LogGroupName', /* required */
    },
  ],
  MetricName: 'IncomingLogEvents',
  Namespace: 'AWS/Logs'
};

cw.listMetrics(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Metrics", JSON.stringify(data.Metrics));
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListMetrics](#) in [AWS SDK for JavaScript API Reference](#).

## Python

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:  
    """Encapsulates Amazon CloudWatch functions."""  
    def __init__(self, cloudwatch_resource):  
        """  
        :param cloudwatch_resource: A Boto3 CloudWatch resource.  
        """  
        self.cloudwatch_resource = cloudwatch_resource  
  
    def list_metrics(self, namespace, name, recent=False):  
        """  
        Gets the metrics within a namespace that have the specified name.  
        If the metric has no dimensions, a single metric is returned.  
        Otherwise, metrics for all dimensions are returned.  
  
        :param namespace: The namespace of the metric.  
        :param name: The name of the metric.  
        :param recent: When True, only metrics that have been active in the last  
                      three hours are returned.  
        :return: An iterator that yields the retrieved metrics.  
        """  
        try:  
            kwargs = {'Namespace': namespace, 'MetricName': name}  
            if recent:  
                kwargs['RecentlyActive'] = 'PT3H' # List past 3 hours only  
                metric_iter = self.cloudwatch_resource.metrics.filter(**kwargs)  
                logger.info("Got metrics for %s.%s.", namespace, name)  
            except ClientError:  
                logger.exception("Couldn't get metrics for %s.%s.", namespace, name)  
                raise  
            else:  
                return metric_iter
```

- For API details, see [ListMetrics](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"The following list-metrics example displays the metrics for Amazon  
Cloudwatch."  
TRY.  
    oo_result = lo_cwt->listmetrics(           " oo_result is returned for  
    testing purpose "  
        iv_namespace = iv_namespace  
    ).  
    DATA(lt_metrics) = oo_result->get_metrics( ).  
    MESSAGE 'Metrics retrieved' TYPE 'I'.  
CATCH /aws1/cx_cwtinparamvalueex .  
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [ListMetrics](#) in *AWS SDK for SAP ABAP API reference*.

## Put a set of data into a CloudWatch metric using an AWS SDK

The following code example shows how to put a set of data into a Amazon CloudWatch metric.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:  
    """Encapsulates Amazon CloudWatch functions."""  
    def __init__(self, cloudwatch_resource):  
        """  
        :param cloudwatch_resource: A Boto3 CloudWatch resource.  
        """  
        self.cloudwatch_resource = cloudwatch_resource  
  
    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):  
        """  
        Sends a set of data to CloudWatch for a metric. All of the data in the set  
        have the same timestamp and unit.  
  
        :param namespace: The namespace of the metric.  
        :param name: The name of the metric.  
        :param timestamp: The UTC timestamp for the metric.  
        :param unit: The unit of the metric.  
        :param data_set: The set of data to send. This set is a dictionary that  
                        contains a list of values and a list of corresponding  
                        counts.  
                        The value and count lists must be the same length.  
        """  
        try:  
            metric = self.cloudwatch_resource.Metric(namespace, name)  
            metric.put_data(  
                Namespace=namespace,  
                MetricData=[{  
                    'MetricName': name,  
                    'Timestamp': timestamp,  
                    'Values': data_set['values'],  
                    'Counts': data_set['counts'],  
                    'Unit': unit}])  
            logger.info("Put data set for metric %s.%s.", namespace, name)  
        except ClientError:  
            logger.exception("Couldn't put data set for metric %s.%s.", namespace,  
                             name)  
            raise
```

- For API details, see [PutMetricData](#) in *AWS SDK for Python (Boto3) API Reference*.

## Put data into a CloudWatch metric using an AWS SDK

The following code examples show how to put data into a Amazon CloudWatch metric.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricDataRequest.h>
#include <iostream>
```

Put data into the metric.

```
Aws::CloudWatch::CloudWatchClient cw;

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("UNIQUE_PAGES");
dimension.SetValue("URLS");

Aws::CloudWatch::Model::MetricDatum datum;
datum.SetMetricName("PAGES_VISITED");
datum.SetUnit(Aws::CloudWatch::Model::StandardUnit::None);
datum.SetValue(data_point);
datum.AddDimensions(dimension);

Aws::CloudWatch::Model::PutMetricDataRequest request;
request.SetNamespace("SITE/TRAFFIC");
request.AddMetricData(datum);

auto outcome = cw.PutMetricData(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to put sample metric data:" <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully put sample metric data" << std::endl;
}
```

- For API details, see [PutMetricData](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void putMetData(CloudWatchClient cw, Double dataPoint ) {

    try {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
```

```
.value("URLS")
.build();

// Set an Instant object.
String time =
ZonedDateTime.now( ZoneOffset.UTC ).format( DateTimeFormatter.ISO_INSTANT );
Instant instant = Instant.parse(time);

MetricDatum datum = MetricDatum.builder()
    .metricName("PAGES_VISITED")
    .unit(StandardUnit.NONE)
    .value(dataPoint)
    .timestamp(instant)
    .dimensions(dimension).build();

PutMetricDataRequest request = PutMetricDataRequest.builder()
    .namespace("SITE/TRAFFIC")
    .metricData(datum).build();

cw.putMetricData(request);

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Successfully put data point %f", dataPoint);
}
```

- For API details, see [PutMetricData](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { PutMetricDataCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = {
  MetricData: [
    {
      MetricName: "PAGES_VISITED",
```

```
Dimensions: [
  {
    Name: "UNIQUE_PAGES",
    Value: "URLS",
  },
],
Unit: "None",
Value: 1.0,
},
],
Namespace: "SITE/TRAFFIC",
};

export const run = async () => {
try {
  const data = await cwClient.send(new PutMetricDataCommand(params));
  console.log("Success", data.$metadata.requestId);
  return data;
} catch (err) {
  console.log("Error", err);
}
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutMetricData](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

// Create parameters JSON for putMetricData
var params = {
  MetricData: [
    {
      MetricName: 'PAGES_VISITED',
      Dimensions: [
        {
          Name: 'UNIQUE_PAGES',
          Value: 'URLS'
        },
      ],
      Unit: 'None',
      Value: 1.0
    },
  ],
  Namespace: 'SITE/TRAFFIC'
};

cw.putMetricData(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  }
});
```

```
    } else {
        console.log("Success", JSON.stringify(data));
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutMetricData](#) in [AWS SDK for JavaScript API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""
    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource

    def put_metric_data(self, namespace, name, value, unit):
        """
        Sends a single data value to CloudWatch for a metric. This metric is given
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
        :param unit: The unit of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{
                    'MetricName': name,
                    'Value': value,
                    'Unit': unit
                }])
            logger.info("Put data for metric %s.%s", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data for metric %s.%s", namespace, name)
            raise
```

- For API details, see [PutMetricData](#) in [AWS SDK for Python \(Boto3\) API Reference](#).

## Scenarios for CloudWatch using AWS SDKs

The following code examples show how to use Amazon CloudWatch with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Get started with CloudWatch alarms using an AWS SDK \(p. 179\)](#)
- [Manage CloudWatch metrics and alarms using an AWS SDK \(p. 180\)](#)

## Get started with CloudWatch alarms using an AWS SDK

The following code example shows how to:

- Create an alarm.
- Disable alarm actions.
- Describe an alarm.
- Delete an alarm.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lt_alarmnames TYPE /aws1/cl_cwtalarmnames_w=>tt_alarmnames.
DATA lo_alarmname TYPE REF TO /aws1/cl_cwtalarmnames_w.

"Create an alarm"
TRY.
    lo_cwt->putmetricalarm(
        iv_alarmname           = iv_alarm_name
        iv_comparisonoperator  = iv_comparison_operator
        iv_evaluationperiods  = iv_evaluation_periods
        iv_metricname          = iv_metric_name
        iv_namespace            = iv_namespace
        iv_statistic             = iv_statistic
        iv_threshold             = iv_threshold
        iv_actionsenabled       = iv_actions_enabled
        iv_alarmdescription     = iv_alarm_description
        iv_unit                  = iv_unit
        iv_period                 = iv_period
        it_dimensions            = it_dimensions
    ).
    MESSAGE 'Alarm created' TYPE 'I'.
    CATCH /aws1/cx_cwtlimitexceededfault.
        MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
ENDTRY.

"Create an ABAP internal table for the alarm created"
CREATE OBJECT lo_alarmname EXPORTING iv_value = iv_alarm_name.
INSERT lo_alarmname INTO TABLE lt_alarmnames.

"Disable alarm actions"
TRY.
    lo_cwt->disablealarmactions(
        it_alarmnames           = lt_alarmnames
    ).
    MESSAGE 'Alarm actions disabled' TYPE 'I'.
```

```
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_disablealarm_exception).
  DATA(lv_disablealarm_error) = |"{ lo_disablealarm_exception->av_err_code }"
- { lo_disablealarm_exception->av_err_msg }|.
  MESSAGE lv_disablealarm_error TYPE 'E'.
ENDTRY.

"Describe alarm using the same ABAP internal table"
TRY.
  oo_result = lo_cwt->describealarms(                               " oo_result is
returned for testing purpose "
    it_alarmnames           = lt_alarmnames
  ).
  MESSAGE 'Alarms retrieved' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_describealarms_exception).
  DATA(lv_describealarms_error) = |"{ lo_describealarms_exception-
>av_err_code }" - { lo_describealarms_exception->av_err_msg }|.
  MESSAGE lv_describealarms_error TYPE 'E'.
ENDTRY.

"Delete alarm"
TRY.
  lo_cwt->deletealarms(
    it_alarmnames = lt_alarmnames
  ).
  MESSAGE 'Alarms deleted' TYPE 'I'.
CATCH /aws1/cx_cwtresourcenotfound .
  MESSAGE 'Resource being access is not found.' TYPE 'E'.
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [DeleteAlarms](#)
  - [DescribeAlarms](#)
  - [DisableAlarmActions](#)
  - [PutMetricAlarm](#)

## Manage CloudWatch metrics and alarms using an AWS SDK

The following code example shows how to:

- Create an alarm that watches a metric.
- Put data into a CloudWatch metric and trigger the alarm.
- Get data from the alarm.
- Delete the alarm.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps CloudWatch operations.

```
from datetime import datetime, timedelta
import logging
from pprint import pprint
import random
```

```
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""
    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource

    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
        :param data_set: The set of data to send. This set is a dictionary that
            contains a list of values and a list of corresponding
        counts.
            The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{
                    'MetricName': name,
                    'Timestamp': timestamp,
                    'Values': data_set['values'],
                    'Counts': data_set['counts'],
                    'Unit': unit}])
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.", namespace,
                             name)
            raise

    def create_metric_alarm(
        self, metric_namespace, metric_name, alarm_name, stat_type, period,
        eval_periods, threshold, comparison_op):
        """
        Creates an alarm that watches a metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
        :param period: The period in which metric data are grouped to calculate
            statistics.
        :param eval_periods: The number of periods that the metric must be over the
            alarm threshold before the alarm is set into an
        alarmed
            state.
        :param threshold: The threshold value to compare against the metric
            statistic.
        :param comparison_op: The comparison operation used to compare the
            threshold
            against the metric.
        """

```

```
:return: The newly created alarm.  
"""  
    try:  
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)  
        alarm = metric.put_alarm(  
            AlarmName=alarm_name,  
            Statistic=stat_type,  
            Period=period,  
            EvaluationPeriods=eval_periods,  
            Threshold=threshold,  
            ComparisonOperator=comparison_op)  
        logger.info(  
            "Added alarm %s to track metric %s.%s.", alarm_name,  
metric_namespace,  
            metric_name)  
    except ClientError:  
        logger.exception(  
            "Couldn't add alarm %s to metric %s.%s", alarm_name,  
metric_namespace,  
            metric_name)  
        raise  
    else:  
        return alarm  
  
def put_metric_data(self, namespace, name, value, unit):  
    """  
        Sends a single data value to CloudWatch for a metric. This metric is given  
        a timestamp of the current UTC time.  
  
        :param namespace: The namespace of the metric.  
        :param name: The name of the metric.  
        :param value: The value of the metric.  
        :param unit: The unit of the metric.  
    """  
    try:  
        metric = self.cloudwatch_resource.Metric(namespace, name)  
        metric.put_data(  
            Namespace=namespace,  
            MetricData=[{  
                'MetricName': name,  
                'Value': value,  
                'Unit': unit  
            }]  
        )  
        logger.info("Put data for metric %s.%s", namespace, name)  
    except ClientError:  
        logger.exception("Couldn't put data for metric %s.%s", namespace, name)  
        raise  
  
    def get_metric_statistics(self, namespace, name, start, end, period,  
stat_types):  
        """  
            Gets statistics for a metric within a specified time span. Metrics are  
            grouped  
            into the specified period.  
  
            :param namespace: The namespace of the metric.  
            :param name: The name of the metric.  
            :param start: The UTC start time of the time span to retrieve.  
            :param end: The UTC end time of the time span to retrieve.  
            :param period: The period, in seconds, in which to group metrics. The  
period  
                must  
                match the granularity of the metric, which depends on  
                the metric's age. For example, metrics that are older than  
                three hours have a one-minute granularity, so the period  
            must
```

```

        be at least 60 and must be a multiple of 60.
:param stat_types: The type of statistics to retrieve, such as average
value
        or maximum value.
:return: The retrieved statistics for the metric.
"""
try:
    metric = self.cloudwatch_resource.Metric(namespace, name)
    stats = metric.get_statistics(
        StartTime=start, EndTime=end, Period=period, Statistics=stat_types)
    logger.info(
        "Got %s statistics for %s.", len(stats['Datapoints']),
        stats['Label'])
except ClientError:
    logger.exception("Couldn't get statistics for %s.%s.", namespace, name)
    raise
else:
    return stats

def get_metric_alarms(self, metric_namespace, metric_name):
"""
Gets the alarms that are currently watching the specified metric.

:param metric_namespace: The namespace of the metric.
:param metric_name: The name of the metric.
:returns: An iterator that yields the alarms.
"""
metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
alarm_iter = metric.alarms.all()
logger.info("Got alarms for metric %s.%s.", metric_namespace, metric_name)
return alarm_iter

def delete_metric_alarms(self, metric_namespace, metric_name):
"""
Deletes all of the alarms that are currently watching the specified metric.

:param metric_namespace: The namespace of the metric.
:param metric_name: The name of the metric.
"""
try:
    metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
    metric.alarms.delete()
    logger.info(
        "Deleted alarms for metric %s.%s.", metric_namespace, metric_name)
except ClientError:
    logger.exception(
        "Couldn't delete alarms for metric %s.%s.", metric_namespace,
        metric_name)
    raise

```

Use the wrapper class to put data in a metric, trigger an alarm that watches the metric, and get data from the alarm.

```

def usage_demo():
    print('*'*88)
    print("Welcome to the Amazon CloudWatch metrics and alarms demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    cw_wrapper = CloudWatchWrapper(boto3.resource('cloudwatch'))

    minutes = 20

```

```

metric_namespace = 'doc-example-metric'
metric_name = 'page_views'
start = datetime.utcnow() - timedelta(minutes=minutes)
print(f"Putting data into metric {metric_namespace}.{metric_name} spanning the
"
      f"last {minutes} minutes.")
for offset in range(0, minutes):
    stamp = start + timedelta(minutes=offset)
    cw_wrapper.put_metric_data_set(
        metric_namespace, metric_name, stamp, 'Count', {
            'values': [
                random.randint(bound, bound * 2)
                for bound in range(offset + 1, offset + 11)],
            'counts': [random.randint(1, offset + 1) for _ in range(10)]
        })
alarm_name = 'high_page_views'
period = 60
eval_periods = 2
print(f"Creating alarm {alarm_name} for metric {metric_name}.")
alarm = cw_wrapper.create_metric_alarm(
    metric_namespace, metric_name, alarm_name, 'Maximum', period, eval_periods,
    100, 'GreaterThanThreshold')
print(f"Alarm ARN is {alarm.alarm_arn}.")
print(f"Current alarm state is: {alarm.state_value}.")

print(f"Sending data to trigger the alarm. This requires data over the
threshold "
      f"for {eval_periods} periods of {period} seconds each.")
while alarm.state_value == 'INSUFFICIENT_DATA':
    print("Sending data for the metric.")
    cw_wrapper.put_metric_data(
        metric_namespace, metric_name, random.randint(100, 200), 'Count')
    alarm.load()
    print(f"Current alarm state is: {alarm.state_value}.")
    if alarm.state_value == 'INSUFFICIENT_DATA':
        print(f"Waiting for {period} seconds...")
        time.sleep(period)
    else:
        print("Wait for a minute for eventual consistency of metric data.")
        time.sleep(period)
        if alarm.state_value == 'OK':
            alarm.load()
            print(f"Current alarm state is: {alarm.state_value}.")

print(f"Getting data for metric {metric_namespace}.{metric_name} during
timespan "
      f"of {start} to {datetime.utcnow()} (times are UTC).")
stats = cw_wrapper.get_metric_statistics(
    metric_namespace, metric_name, start, datetime.utcnow(), 60,
    ['Average', 'Minimum', 'Maximum'])
print(f"Got {len(stats['Datapoints'])} data points for metric "
      f"{metric_namespace}.{metric_name}.")
pprint(sorted(stats['Datapoints'], key=lambda x: x['Timestamp']))

print(f"Getting alarms for metric {metric_name}.")
alarms = cw_wrapper.get_metric_alarms(metric_namespace, metric_name)
for alarm in alarms:
    print(f"Alarm {alarm.name} is currently in state {alarm.state_value}.")

print(f"Deleting alarms for metric {metric_name}.")
cw_wrapper.delete_metric_alarms(metric_namespace, metric_name)

print("Thanks for watching!")
print('*'*88)

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [DeleteAlarms](#)
  - [DescribeAlarmsForMetric](#)
  - [DisableAlarmActions](#)
  - [EnableAlarmActions](#)
  - [GetMetricStatistics](#)
  - [ListMetrics](#)
  - [PutMetricAlarm](#)
  - [PutMetricData](#)

## Code examples for CloudWatch Events using AWS SDKs

The following code examples show how to use Amazon CloudWatch Events with an AWS software development kit (SDK).

### Code examples

- [Actions for CloudWatch Events using AWS SDKs \(p. 185\)](#)
  - [Add a Lambda function target using an AWS SDK \(p. 185\)](#)
  - [Create a CloudWatch Events scheduled rule using an AWS SDK \(p. 187\)](#)
  - [Send CloudWatch Events events using an AWS SDK \(p. 189\)](#)

## Actions for CloudWatch Events using AWS SDKs

The following code examples show how to use Amazon CloudWatch Events with AWS SDKs. Each example calls an individual service function.

### Examples

- [Add a Lambda function target using an AWS SDK \(p. 185\)](#)
- [Create a CloudWatch Events scheduled rule using an AWS SDK \(p. 187\)](#)
- [Send CloudWatch Events events using an AWS SDK \(p. 189\)](#)

## Add a Lambda function target using an AWS SDK

The following code examples show how to add an AWS Lambda function target to an Amazon CloudWatch Events event.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void putCWTTargets(CloudWatchEventsClient cwe, String ruleName,
String functionArn, String targetId ) {

    try {
        Target target = Target.builder()
            .arn(functionArn)
            .id(targetId)
            .build();

        PutTargetsRequest request = PutTargetsRequest.builder()
            .targets(target)
            .rule(ruleName)
            .build();

        cwe.putTargets(request);
        System.out.printf(
            "Successfully created CloudWatch events target for rule %s",
            ruleName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutTargets](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cweClient = new CloudWatchEventsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { PutTargetsCommand } from "@aws-sdk/client-cloudwatch-events";
import { cweClient } from "./libs/cloudWatchEventsClient.js";

// Set the parameters
export const params = {
  Rule: "DEMO_EVENT",
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN", //LAMBDA_FUNCTION_ARN
      Id: "myCloudWatchEventsTarget",
    },
  ],
}
```

```
    ],
};

export const run = async () => {
  try {
    const data = await cweClient.send(new PutTargetsCommand(params));
    console.log("Success, target added; requestID: ", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutTargets](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({apiVersion: '2015-10-07'});

var params = {
  Rule: 'DEMO_EVENT',
  Targets: [
    {
      Arn: 'LAMBDA_FUNCTION_ARN',
      Id: 'myCloudWatchEventsTarget',
    }
  ]
};

cwevents.putTargets(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutTargets](#) in [AWS SDK for JavaScript API Reference](#).

## Create a CloudWatch Events scheduled rule using an AWS SDK

The following code examples show how to create an Amazon CloudWatch Events scheduled rule.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void putCWRule(CloudWatchEventsClient cwe, String ruleName,
String roleArn) {

    try {
        PutRuleRequest request = PutRuleRequest.builder()
            .name(ruleName)
            .roleArn(roleArn)
            .scheduleExpression("rate(5 minutes)")
            .state(RuleState.ENABLED)
            .build();

        PutRuleResponse response = cwe.putRule(request);
        System.out.printf(
            "Successfully created CloudWatch events rule %s with arn %s",
            roleArn, response.ruleArn());

    } catch (
        CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutRule](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cweClient = new CloudWatchEventsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { PutRuleCommand } from "@aws-sdk/client-cloudwatch-events";
import { cweClient } from "./libs/cloudWatchEventsClient.js";
```

```
// Set the parameters
export const params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN", //IAM_ROLE_ARN
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

export const run = async () => {
  try {
    const data = await cweClient.send(new PutRuleCommand(params));
    console.log("Success, scheduled rule created; Rule ARN:", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutRule in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({apiVersion: '2015-10-07'});

var params = {
  Name: 'DEMO_EVENT',
  RoleArn: 'IAM_ROLE_ARN',
  ScheduleExpression: 'rate(5 minutes)',
  State: 'ENABLED'
};

cwevents.putRule(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.RuleArn);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutRule in AWS SDK for JavaScript API Reference](#).

## Send CloudWatch Events events using an AWS SDK

The following code examples show how to send Amazon CloudWatch Events events.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void putCWEVENTS(CloudWatchEventsClient cwe, String resourceArn ) {
    try {
        final String EVENT_DETAILS =
            "{ \"key1\": \"value1\", \"key2\": \"value2\" }";

        PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()
            .detail(EVENT_DETAILS)
            .detailType("sampleSubmitted")
            .resources(resourceArn)
            .source("aws-sdk-java-cloudwatch-example")
            .build();

        PutEventsRequest request = PutEventsRequest.builder()
            .entries(requestEntry)
            .build();

        cwe.putEvents(request);
        System.out.println("Successfully put CloudWatch event");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutEvents](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cweClient = new CloudWatchEventsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { PutEventsCommand } from "@aws-sdk/client-cloudwatch-events";
```

```
import { cweClient } from "./libs/cloudWatchEventsClient.js";

// Set the parameters
export const params = {
  Entries: [
    {
      Detail: '{ "key1": "value1", "key2": "value2" }',
      DetailType: "appRequestSubmitted",
      Resources: [
        "RESOURCE_ARN", //RESOURCE_ARN
      ],
      Source: "com.company.app",
    },
  ],
};

export const run = async () => {
  try {
    const data = await cweClient.send(new PutEventsCommand(params));
    console.log("Success, event sent; requestID:", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutEvents](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({apiVersion: '2015-10-07'});

var params = {
  Entries: [
    {
      Detail: '{ \"key1\": \"value1\", \"key2\": \"value2\" }',
      DetailType: 'appRequestSubmitted',
      Resources: [
        'RESOURCE_ARN',
      ],
      Source: 'com.company.app'
    }
  ]
};

cwevents.putEvents(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
```

```
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutEvents](#) in [AWS SDK for JavaScript API Reference](#).

## Code examples for CloudWatch Logs using AWS SDKs

The following code examples show how to use Amazon CloudWatch Logs with an AWS software development kit (SDK).

### Code examples

- [Actions for CloudWatch Logs using AWS SDKs \(p. 192\)](#)
  - Associate an AWS KMS key with a CloudWatch Logs log group using an AWS SDK (p. 193)
  - Cancel a CloudWatch Logs export task using an AWS SDK (p. 194)
  - Create a CloudWatch Logs log group using an AWS SDK (p. 194)
  - Create a CloudWatch Logs log stream using an AWS SDK (p. 195)
  - Create a CloudWatch Logs subscription filter using an AWS SDK (p. 196)
  - Create a CloudWatch Logs export task using an AWS SDK (p. 199)
  - Delete a CloudWatch Logs log group using an AWS SDK (p. 200)
  - Delete a CloudWatch Logs subscription filter using an AWS SDK (p. 201)
  - Describe CloudWatch Logs subscription filters using an AWS SDK (p. 204)
  - Describe CloudWatch Logs export tasks using an AWS SDK (p. 208)
  - Describe CloudWatch Logs log groups using an AWS SDK (p. 209)
- [Cross-service examples for CloudWatch Logs using AWS SDKs \(p. 210\)](#)
  - Use scheduled events to invoke a Lambda function (p. 210)

## Actions for CloudWatch Logs using AWS SDKs

The following code examples show how to use Amazon CloudWatch Logs with AWS SDKs. Each example calls an individual service function.

### Examples

- [Associate an AWS KMS key with a CloudWatch Logs log group using an AWS SDK \(p. 193\)](#)
- [Cancel a CloudWatch Logs export task using an AWS SDK \(p. 194\)](#)
- [Create a CloudWatch Logs log group using an AWS SDK \(p. 194\)](#)
- [Create a CloudWatch Logs log stream using an AWS SDK \(p. 195\)](#)
- [Create a CloudWatch Logs subscription filter using an AWS SDK \(p. 196\)](#)
- [Create a CloudWatch Logs export task using an AWS SDK \(p. 199\)](#)
- [Delete a CloudWatch Logs log group using an AWS SDK \(p. 200\)](#)
- [Delete a CloudWatch Logs subscription filter using an AWS SDK \(p. 201\)](#)
- [Describe CloudWatch Logs subscription filters using an AWS SDK \(p. 204\)](#)
- [Describe CloudWatch Logs export tasks using an AWS SDK \(p. 208\)](#)
- [Describe CloudWatch Logs log groups using an AWS SDK \(p. 209\)](#)

## Associate an AWS KMS key with a CloudWatch Logs log group using an AWS SDK

The following code example shows how to associate an AWS KMS key with an existing CloudWatch Logs log group.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to associate an AWS Key Management Service (AWS KMS) key with
/// an Amazon CloudWatch Logs log group. The example was created using the
/// AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class AssociateKmsKey
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();

        string kmsKeyId = "arn:aws:kms:us-west-2:<account-
number>:key/7c9ecc2-38cb-4c4f-9db3-766ee8dd3ad4";
        string groupName = "cloudwatchlogs-example-loggroup";

        var request = new AssociateKmsKeyRequest
        {
            KmsKeyId = kmsKeyId,
            LogGroupName = groupName,
        };

        var response = await client.AssociateKmsKeyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully associated KMS key ID: {kmsKeyId}
with log group: {groupName}.");
        }
        else
        {
            Console.WriteLine("Could not make the association between:
{kmsKeyId} and {groupName}.");
        }
    }
}
```

- For API details, see [AssociateKmsKey](#) in *AWS SDK for .NET API Reference*.

## Cancel a CloudWatch Logs export task using an AWS SDK

The following code example shows how to cancel an existing CloudWatch Logs export task.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to cancel an Amazon CloudWatch Logs export task. The example
/// uses the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class CancelExportTask
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string taskId = "exampleTaskId";

        var request = new CancelExportTaskRequest
        {
            TaskId = taskId,
        };

        var response = await client.CancelExportTaskAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"'{taskId}' successfully canceled.");
        }
        else
        {
            Console.WriteLine($"'{taskId}' could not be canceled.");
        }
    }
}
```

- For API details, see [CancelExportTask](#) in *AWS SDK for .NET API Reference*.

## Create a CloudWatch Logs log group using an AWS SDK

The following code example shows how to create a new CloudWatch Logs log group.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Amazon CloudWatch Logs log group. The example
/// was created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class CreateLogGroup
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();

        string logGroupName = "cloudwatchlogs-example-loggroup";

        var request = new CreateLogGroupRequest
        {
            LogGroupName = logGroupName,
        };

        var response = await client.CreateLogGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully create log group with ID:
{logGroupName}.");
        }
        else
        {
            Console.WriteLine("Could not create log group.");
        }
    }
}
```

- For API details, see [CreateLogGroup](#) in *AWS SDK for .NET API Reference*.

## Create a CloudWatch Logs log stream using an AWS SDK

The following code example shows how to create a new CloudWatch Logs log stream.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Amazon CloudWatch Logs stream for a CloudWatch
/// log group. The example was created using the AWS SDK for .NET version
/// 3.7 and .NET Core 5.0.
/// </summary>
public class CreateLogStream
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string logGroupName = "cloudwatchlogs-example-loggroup";
        string logStreamName = "cloudwatchlogs-example-logstream";

        var request = new CreateLogStreamRequest
        {
            LogGroupName = logGroupName,
            LogStreamName = logStreamName,
        };

        var response = await client.CreateLogStreamAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"{logStreamName} successfully created for
{logGroupName}.");
        }
        else
        {
            Console.WriteLine("Could not create stream.");
        }
    }
}
```

- For API details, see [CreateLogStream](#) in [AWS SDK for .NET API Reference](#).

## Create a CloudWatch Logs subscription filter using an AWS SDK

The following code examples show how to create an Amazon CloudWatch Logs subscription filter.

C++

**SDK for C++**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/PutSubscriptionFilterRequest.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Create the subscription filter.

```
Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
Aws::CloudWatchLogs::Model::PutSubscriptionFilterRequest request;
request.SetFilterName(filter_name);
request.SetFilterPattern(filter_pattern);
request.SetLogGroupName(log_group);
request.SetDestinationArn(dest_arn);
auto outcome = cwl.PutSubscriptionFilter(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch logs subscription filter "
        << filter_name << ":" << outcome.GetError().GetMessage() <<
        std::endl;
}
else
{
    std::cout << "Successfully created CloudWatch logs subscription " <<
        "filter " << filter_name << std::endl;
}
```

- For API details, see [PutSubscriptionFilter in AWS SDK for C++ API Reference](#).

Java

**SDK for Java 2.x**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void putSubFilters(CloudWatchLogsClient cwl,
                                  String filter,
                                  String pattern,
                                  String logGroup,
                                  String functionArn) {

    try {
        PutSubscriptionFilterRequest request =
PutSubscriptionFilterRequest.builder()
            .filterName(filter)
            .filterPattern(pattern)
```

```
.logGroupName(logGroup)
.destinationArn(functionArn)
.build();

cwl.putSubscriptionFilter(request);
System.out.printf(
    "Successfully created CloudWatch logs subscription filter %s",
    filter);

} catch (CloudWatchLogsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [PutSubscriptionFilter](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch Logs service client object.
export const cwlClient = new CloudWatchLogsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {
  PutSubscriptionFilterCommand,
} from "@aws-sdk/client-cloudwatch-logs";
import { cwlClient } from "./libs/cloudWatchLogsClient.js";

// Set the parameters
export const params = {
  destinationArn: "LAMBDA_FUNCTION_ARN", //LAMBDA_FUNCTION_ARN
  filterName: "FILTER_NAME", //FILTER_NAME
  filterPattern: "ERROR",
  logGroupName: "LOG_GROUP", //LOG_GROUP
};

export const run = async () => {
  try {
    const data = await cwlClient.send(new PutSubscriptionFilterCommand(params));
    console.log("Success", data.subscriptionFilters);
    return data; //For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
```

```
// Uncomment this line to run execution within this file.  
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutSubscriptionFilter](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create the CloudWatchLogs service object  
var cwl = new AWS.CloudWatchLogs({apiVersion: '2014-03-28'});  
  
var params = {  
    destinationArn: 'LAMBDA_FUNCTION_ARN',  
    filterName: 'FILTER_NAME',  
    filterPattern: 'ERROR',  
    logGroupName: 'LOG_GROUP',  
};  
  
cwl.putSubscriptionFilter(params, function(err, data) {  
    if (err) {  
        console.log("Error", err);  
    } else {  
        console.log("Success", data);  
    }  
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutSubscriptionFilter](#) in [AWS SDK for JavaScript API Reference](#).

## Create a CloudWatch Logs export task using an AWS SDK

The following code example shows how to create a new CloudWatch Logs export task.

### .NET

#### AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Threading.Tasks;  
using Amazon.CloudWatchLogs;  
using Amazon.CloudWatchLogs.Model;  
  
/// <summary>
```

```
/// Shows how to create an Export Task to export the contents of the Amazon
/// CloudWatch Logs to the specified Amazon Simple Storage Service (Amazon S3)
/// bucket. The example was created with the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
/// </summary>
public class CreateExportTask
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string taskName = "export-task-example";
        string logGroupName = "cloudwatchlogs-example-loggroup";
        string destination = "doc-example-bucket";
        var fromTime = 1437584472382;
        var toTime = 1437584472833;

        var request = new CreateExportTaskRequest
        {
            From = fromTime,
            To = toTime,
            TaskName = taskName,
            LogGroupName = logGroupName,
            Destination = destination,
        };

        var response = await client.CreateExportTaskAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"The task, {taskName} with ID: " +
                $"{response.TaskId} has been created
successfully.");
        }
    }
}
```

- For API details, see [CreateExportTask](#) in *AWS SDK for .NET API Reference*.

## Delete a CloudWatch Logs log group using an AWS SDK

The following code example shows how to delete an existing CloudWatch Logs log group.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;
```

```
/// <summary>
/// Uses the Amazon CloudWatch Logs Service to delete an existing
/// CloudWatch Logs log group. The example was created using the
/// AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DeleteLogGroup
{
    public static async Task Main()
    {
        var client = new AmazonCloudWatchLogsClient();
        string logGroupName = "cloudwatchlogs-example-loggroup";

        var request = new DeleteLogGroupRequest
        {
            LogGroupName = logGroupName,
        };

        var response = await client.DeleteLogGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully deleted CloudWatch log group,
{logGroupName}.");
        }
    }
}
```

- For API details, see [DeleteLogGroup](#) in *AWS SDK for .NET API Reference*.

## Delete a CloudWatch Logs subscription filter using an AWS SDK

The following code examples show how to delete an Amazon CloudWatch Logs subscription filter.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>
#include <iostream>
```

Delete the subscription filter.

```
Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
Aws::CloudWatchLogs::Model::DeleteSubscriptionFilterRequest request;
request.SetFilterName(filter_name);
request.SetLogGroupName(log_group);
```

```
auto outcome = cwl.DeleteSubscriptionFilter(request);
if (!outcome.IsSuccess()) {
    std::cout << "Failed to delete CloudWatch log subscription filter "
        << filter_name << ":" << outcome.GetError().GetMessage() <<
        std::endl;
} else {
    std::cout << "Successfully deleted CloudWatch logs subscription " <<
        "filter " << filter_name << std::endl;
}
```

- For API details, see [DeleteSubscriptionFilter](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteSubFilter(CloudWatchLogsClient logs, String filter,
String logGroup) {

    try {
        DeleteSubscriptionFilterRequest request =
DeleteSubscriptionFilterRequest.builder()
            .filterName(filter)
            .logGroupName(logGroup)
            .build();

        logs.deleteSubscriptionFilter(request);
        System.out.printf("Successfully deleted CloudWatch logs subscription
filter %s", filter);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteSubscriptionFilter](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch Logs service client object.
```

```
export const cwlClient = new CloudWatchLogsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {
  DeleteSubscriptionFilterCommand
} from "@aws-sdk/client-cloudwatch-logs";
import { cwlClient } from "./libs/cloudWatchLogsClient.js";

// Set the parameters
export const params = {
  filterName: "FILTER", //FILTER
  logGroupName: "LOG_GROUP", //LOG_GROUP
};

export const run = async () => {
  try {
    const data = await cwlClient.send(
      new DeleteSubscriptionFilterCommand(params)
    );
    console.log(
      "Success, subscription filter deleted",
      data
    );
    return data; //For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteSubscriptionFilter](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({apiVersion: '2014-03-28'});

var params = {
  filterName: 'FILTER',
  logGroupName: 'LOG_GROUP'
};

cwl.deleteSubscriptionFilter(params, function(err, data) {
  if (err) {
    console.log("Error", err);
```

```
    } else {
        console.log("Success", data);
    });
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteSubscriptionFilter](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteSubFilter(filter: String?, logGroup: String?) {

    val request = DeleteSubscriptionFilterRequest {
        filterName = filter
        logGroupName = logGroup
    }

    CloudWatchLogsClient { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
$filter")
    }
}
```

- For API details, see [DeleteSubscriptionFilter](#) in [AWS SDK for Kotlin API reference](#).

## Describe CloudWatch Logs subscription filters using an AWS SDK

The following code examples show how to describe Amazon CloudWatch Logs existing subscription filters.

### C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
```

```
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DescribeSubscriptionFiltersRequest.h>
#include <aws/logs/model/DescribeSubscriptionFiltersResult.h>
#include <iostream>
#include <iomanip>
```

List the subscription filters.

```
Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
Aws::CloudWatchLogs::Model::DescribeSubscriptionFiltersRequest request;
request.SetLogGroupName(log_group);
request.SetLimit(1);

bool done = false;
bool header = false;
while (!done) {
    auto outcome = cwl.DescribeSubscriptionFilters(
        request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to describe CloudWatch subscription filters "
        << "for log group " << log_group << ":" <<
        outcome.GetError().GetMessage() << std::endl;
        break;
    }

    if (!header) {
        std::cout << std::left << std::setw(32) << "Name" <<
            std::setw(64) << "FilterPattern" << std::setw(64) <<
            "DestinationArn" << std::endl;
        header = true;
    }

    const auto &filters = outcome.GetResult().GetSubscriptionFilters();
    for (const auto &filter : filters) {
        std::cout << std::left << std::setw(32) <<
            filter.GetFilterName() << std::setw(64) <<
            filter.GetFilterPattern() << std::setw(64) <<
            filter.GetDestinationArn() << std::endl;
    }

    const auto &next_token = outcome.GetResult().GetNextToken();
    request.SetNextToken(next_token);
    done = next_token.empty();
}
```

- For API details, see [DescribeSubscriptionFilters](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeFilters(CloudWatchLogsClient logs, String logGroup)
{
    try {
```

```
boolean done = false;
String newToken = null;

while(!done) {
    DescribeSubscriptionFiltersResponse response;
    if (newToken == null) {
        DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
            .logGroupName(logGroup)
            .limit(1).build();

        response = logs.describeSubscriptionFilters(request);
    } else {
        DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
            .nextToken(newToken)
            .logGroupName(logGroup)
            .limit(1).build();
        response = logs.describeSubscriptionFilters(request);
    }

    for(SubscriptionFilter filter : response.subscriptionFilters()) {
        System.out.printf("Retrieved filter with name %s, " + "pattern
%s " + "and destination arn %s",
                    filter.filterName(),
                    filter.filterPattern(),
                    filter.destinationArn());
    }

    if(response.nextToken() == null) {
        done = true;
    } else {
        newToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Done");
}
```

- For API details, see [DescribeSubscriptionFilters](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch Logs service client object.
export const cwlClient = new CloudWatchLogsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { DescribeSubscriptionFiltersCommand } from "@aws-sdk/client-cloudwatch-logs";
import { cwlClient } from "./libs/cloudWatchLogsClient.js";

// Set the parameters
export const params = {
  logGroupName: "GROUP_NAME", //GROUP_NAME
  limit: 5
};

export const run = async () => {
  try {
    const data = await cwlClient.send(
      new DescribeSubscriptionFiltersCommand(params)
    );
    console.log("Success", data.subscriptionFilters);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeSubscriptionFilters](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({apiVersion: '2014-03-28'});

var params = {
  logGroupName: 'GROUP_NAME',
  limit: 5
};

cwl.describeSubscriptionFilters(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.subscriptionFilters);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeSubscriptionFilters](#) in [AWS SDK for JavaScript API Reference](#).

Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeFilters(logGroup: String) {

    val request = DescribeSubscriptionFiltersRequest {
        logGroupName = logGroup
        limit = 1
    }

    CloudWatchLogsClient { region = "us-west-2" }.use { cwlClient ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name ${filter.filterName} pattern
${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- For API details, see [DescribeSubscriptionFilters](#) in *AWS SDK for Kotlin API reference*.

## Describe CloudWatch Logs export tasks using an AWS SDK

The following code example shows how to describe CloudWatch Logs export tasks.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to retrieve a list of information about Amazon CloudWatch
/// Logs export tasks. The example was created using the AWS SDK for .NET
/// version 3.7 and .NET Core 5.0.
/// </summary>
public class DescribeExportTasks
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
```

```
// as the default user on this system. If you need to use a
// different AWS Region, pass it as a parameter to the client
// constructor.
var client = new AmazonCloudWatchLogsClient();

var request = new DescribeExportTasksRequest
{
    Limit = 5,
};

var response = new DescribeExportTasksResponse();

do
{
    response = await client.DescribeExportTasksAsync(request);
    response.ExportTasks.ForEach(t =>
    {
        Console.WriteLine($"'{t.TaskName}' with ID: {t.TaskId} has
status: {t.Status}");
    });
}
while (response.NextToken is not null);
}
```

- For API details, see [DescribeExportTasks](#) in *AWS SDK for .NET API Reference*.

## Describe CloudWatch Logs log groups using an AWS SDK

The following code example shows how to describe CloudWatch Logs log groups.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Retrieves information about existing Amazon CloudWatch Logs log groups
/// and displays the information on the console. The example was created
/// using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DescribeLogGroups
{
    public static async Task Main()
    {
        // Creates a CloudWatch Logs client using the default
        // user. If you need to work with resources in another
        // AWS Region than the one defined for the default user,
        // pass the AWS Region as a parameter to the client constructor.
        var client = new AmazonCloudWatchLogsClient();

        var request = new DescribeLogGroupsRequest
```

```
{  
    Limit = 5,  
};  
  
var response = await client.DescribeLogGroupsAsync(request);  
  
if (response.LogGroups.Count > 0)  
{  
    do  
    {  
        response.LogGroups.ForEach(lg =>  
        {  
            Console.WriteLine($"[lg.LogGroupName] is associated with  
the key: {lg.KmsKeyId}.");  
            Console.WriteLine($"Created on:  
{lg.CreationTime.Date.Date}");  
            Console.WriteLine($"Date for this group will be stored for:  
{lg.RetentionInDays} days.\n");  
        });  
    }  
    while (response.NextToken is not null);  
}  
}  
}
```

- For API details, see [DescribeLogGroups](#) in *AWS SDK for .NET API Reference*.

## Cross-service examples for CloudWatch Logs using AWS SDKs

The following code examples show how to use Amazon CloudWatch Logs with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Use scheduled events to invoke a Lambda function \(p. 210\)](#)

## Use scheduled events to invoke a Lambda function

The following code examples show how to create an AWS Lambda function invoked by an Amazon EventBridge scheduled event.

### Python

#### SDK for Python (Boto3)

This example shows how to register an AWS Lambda function as the target of a scheduled Amazon EventBridge event. The Lambda handler writes a friendly message and the full event data to Amazon CloudWatch Logs for later retrieval.

- Deploys a Lambda function.
- Creates an EventBridge scheduled event and makes the Lambda function the target.
- Grants permission to let EventBridge invoke the Lambda function.
- Prints the latest data from CloudWatch Logs to show the result of the scheduled invocations.
- Cleans up all resources created during the demo.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- CloudWatch Logs
- EventBridge
- Lambda

## Code examples for Amazon Cognito Identity using AWS SDKs

The following code examples show how to use Amazon Cognito Identity with an AWS software development kit (SDK).

#### Code examples

- [Actions for Amazon Cognito Identity using AWS SDKs \(p. 211\)](#)
  - Create a new Amazon Cognito Identity with a specified name (p. 211)
  - List Amazon Cognito Identity pools (p. 212)
- [Cross-service examples for Amazon Cognito Identity using AWS SDKs \(p. 213\)](#)
  - Build an Amazon Transcribe app (p. 214)
  - Create an Amazon Textract explorer application (p. 214)

## Actions for Amazon Cognito Identity using AWS SDKs

The following code examples show how to use Amazon Cognito Identity with AWS SDKs. Each example calls an individual service function.

#### Examples

- [Create a new Amazon Cognito Identity with a specified name \(p. 211\)](#)
- [List Amazon Cognito Identity pools \(p. 212\)](#)

## Create a new Amazon Cognito Identity with a specified name

The following code example shows how to find the Amazon Cognito Identity with the given name, creating it if it's not found.

Swift

#### SDK for Swift

##### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a new identity pool.

```
func createIdentityPool(name: String) async throws -> String? {
```

```
let cognitoInputCall = CreateIdentityPoolInput(developerProviderName:  
"com.exampleco.CognitoIdentityDemo",  
                                              identityPoolName: name)  
  
do {  
    let result = try await cognitoIdentityClient.createIdentityPool(input:  
cognitoInputCall)  
    guard let poolId = result.identityPoolId else {  
        return nil  
    }  
  
    return poolId  
  
} catch {  
    print("ERROR: ", dump(error, name: "Error attempting to create the  
identity pool"))  
}  
  
return nil  
}
```

- For more information, see [AWS SDK for Swift developer guide](#).
- For API details, see the following topics in *AWS SDK for Swift API reference*.
  - [createIdentityPool](#)
  - [listIdentityPools](#)

## List Amazon Cognito Identity pools

The following code example shows how to get a list of Amazon Cognito Identity pools.

Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Find the ID of an identity pool given its name.

```
func getIdentityPoolID(name: String) async throws -> String? {  
    var token: String? = nil  
  
    // Iterate over the identity pools until a match is found.  
    repeat {  
        /// `token` is a value returned by `ListIdentityPools()` if the  
        returned list  
        /// of identity pools is only a partial list. You use the `token` to  
        tell Cognito that  
        /// you want to continue where you left off previously; specifying  
        `nil` or not providing  
        /// it means "start at the beginning."  
  
        let listPoolsInput = ListIdentityPoolsInput(maxResults: 25, nextToken:  
token)  
  
        /// Read pages of identity pools from Cognito until one is found
```

```
    /// whose name matches the one specified in the `name` parameter.  
    /// Return the matching pool's ID. Each time we ask for the next  
    /// page of identity pools, we pass in the token given by the  
    /// previous page.  
  
    do {  
        let output = try await  
cognitoIdentityClient.listIdentityPools(input: listPoolsInput)  
  
        if let identityPools = output.identityPools {  
            for pool in identityPools {  
                if pool.identityPoolName == name {  
                    return pool.identityPoolId!  
                }  
            }  
        }  
  
        token = output.nextToken  
    } catch {  
        print("ERROR: ", dump(error, name: "Trying to get list of identity  
pools"))  
    }  
} while token != nil  
  
return nil  
}
```

Get the ID of an existing identity pool or create it if it doesn't already exist.

```
public func getOrCreateIdentityPoolID(name: String) async throws -> String? {  
    // See if the pool already exists  
  
    do {  
        guard let poolId = try await self.getIdentityPoolID(name: name) else {  
            let poolId = try await self.createIdentityPool(name: name)  
            return poolId  
        }  
  
        return poolId  
  
    } catch {  
        print("ERROR: ", dump(error, name: "Trying to get the identity pool  
ID"))  
        return nil  
    }  
}
```

- For more information, see [AWS SDK for Swift developer guide](#).
- For API details, see the following topics in *AWS SDK for Swift API reference*.
  - [creatIdentityPool](#)
  - [listIdentityPools](#)

## Cross-service examples for Amazon Cognito Identity using AWS SDKs

The following code examples show how to use Amazon Cognito Identity with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

## Examples

- [Build an Amazon Transcribe app \(p. 214\)](#)
- [Create an Amazon Textract explorer application \(p. 214\)](#)

## Build an Amazon Transcribe app

The following code example shows how to use Amazon Transcribe to transcribe and display voice recordings in the browser.

JavaScript

### SDK for JavaScript V3

Create an app that uses Amazon Transcribe to transcribe and display voice recordings in the browser. The app uses two Amazon Simple Storage Service (Amazon S3) buckets, one to host the application code, and another to store transcriptions. The app uses an Amazon Cognito user pool to authenticate your users. Authenticated users have AWS Identity and Access Management (IAM) permissions to access the required AWS services.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

### Services used in this example

- Amazon Cognito Identity
- Amazon S3
- Amazon Transcribe

## Create an Amazon Textract explorer application

The following code examples show how to explore Amazon Textract output through an interactive application.

JavaScript

### SDK for JavaScript V3

Shows how to use the AWS SDK for JavaScript to build a React application that uses Amazon Textract to extract data from a document image and display it in an interactive web page. This example runs in a web browser and requires an authenticated Amazon Cognito identity for credentials. It uses Amazon Simple Storage Service (Amazon S3) for storage, and for notifications it polls an Amazon Simple Queue Service (Amazon SQS) queue that is subscribed to an Amazon Simple Notification Service (Amazon SNS) topic.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS

- Amazon Textract

## Code examples for Amazon Cognito Identity Provider using AWS SDKs

The following code examples show how to use Amazon Cognito Identity Provider with an AWS software development kit (SDK).

### Code examples

- Actions for Amazon Cognito Identity Provider using AWS SDKs (p. 215)
  - Confirm an Amazon Cognito user using an AWS SDK (p. 216)
  - Confirm an MFA device for tracking by Amazon Cognito using an AWS SDK (p. 219)
  - Get a token to associate an MFA application with an Amazon Cognito user using an AWS SDK (p. 221)
  - Get information about an Amazon Cognito user using an AWS SDK (p. 224)
  - List the Amazon Cognito user pools using an AWS SDK (p. 227)
  - List Amazon Cognito users using an AWS SDK (p. 228)
  - Resend an Amazon Cognito confirmation code using an AWS SDK (p. 230)
  - Respond to Amazon Cognito SRP authentication challenges using an AWS SDK (p. 233)
  - Respond to an Amazon Cognito authentication challenge using an AWS SDK (p. 236)
  - Sign up a user with Amazon Cognito using an AWS SDK (p. 240)
  - Start authentication with a device tracked by Amazon Cognito using an AWS SDK (p. 244)
  - Start authentication with Amazon Cognito and administrator credentials using an AWS SDK (p. 247)
  - Verify an MFA application with an Amazon Cognito user using an AWS SDK (p. 250)
- Scenarios for Amazon Cognito Identity Provider using AWS SDKs (p. 253)
  - Sign up a user with an Amazon Cognito user pool that requires MFA using an AWS SDK (p. 254)

## Actions for Amazon Cognito Identity Provider using AWS SDKs

The following code examples show how to use Amazon Cognito Identity Provider with AWS SDKs. Each example calls an individual service function.

### Examples

- Confirm an Amazon Cognito user using an AWS SDK (p. 216)
- Confirm an MFA device for tracking by Amazon Cognito using an AWS SDK (p. 219)
- Get a token to associate an MFA application with an Amazon Cognito user using an AWS SDK (p. 221)
- Get information about an Amazon Cognito user using an AWS SDK (p. 224)
- List the Amazon Cognito user pools using an AWS SDK (p. 227)
- List Amazon Cognito users using an AWS SDK (p. 228)
- Resend an Amazon Cognito confirmation code using an AWS SDK (p. 230)
- Respond to Amazon Cognito SRP authentication challenges using an AWS SDK (p. 233)
- Respond to an Amazon Cognito authentication challenge using an AWS SDK (p. 236)

- Sign up a user with Amazon Cognito using an AWS SDK (p. 240)
- Start authentication with a device tracked by Amazon Cognito using an AWS SDK (p. 244)
- Start authentication with Amazon Cognito and administrator credentials using an AWS SDK (p. 247)
- Verify an MFA application with an Amazon Cognito user using an AWS SDK (p. 250)

## Confirm an Amazon Cognito user using an AWS SDK

The following code examples show how to confirm an Amazon Cognito user.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Confirms that a user has been signed up successfully.
///</summary>
///<param name="identityProviderClient">An initialized Identity
/// Provider client object.</param>
///<param name="clientId">The client Id of the application associated
/// with the user pool.</param>
///<param name="code">The code sent by the authentication provider
/// to confirm a user's membership in the pool.</param>
///<param name="userName">The user to confirm.</param>
///<returns>A Boolean value indicating the success of the confirmation
/// operation.</returns>
public static async Task<bool> ConfirmSignUp(
    AmazonCognitoIdentityProviderClient identityProviderClient,
    string clientId,
    string code,
    string userName)
{
    var signUpRequest = new ConfirmSignUpRequest
    {
        ClientId = clientId,
        ConfirmationCode = code,
        Username = userName,
    };

    var response = await
identityProviderClient.ConfirmSignUpAsync(signUpRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{userName} was confirmed");
        return true;
    }
    else
    {
        return false;
    }
}
```

- For API details, see [ConfirmSignUp](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code, String userName) {

    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ConfirmSignUp](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const confirmSignUp = async ({ clientId, username, code }) => {
    const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

    const command = new ConfirmSignUpCommand({
        ClientId: clientId,
        Username: username,
        ConfirmationCode: code,
    });

    return client.send(command);
};
```

- For API details, see [ConfirmSignUp](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun confirmSignUp(clientIdVal: String?, codeVal: String?, userNameVal: String?) {  
  
    val signUpRequest = ConfirmSignUpRequest {  
        clientId = clientIdVal  
        confirmationCode = codeVal  
        username = userNameVal  
    }  
  
    CognitoIdentityProviderClient { region = "us-east-1" }.use {  
        identityProviderClient ->  
            identityProviderClient.confirmSignUp(signUpRequest)  
            println("$userNameVal was confirmed")  
    }  
}
```

- For API details, see [ConfirmSignUp](#) in *AWS SDK for Kotlin API reference*.

**Python**

**SDK for Python (Boto3)**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:  
    """Encapsulates Amazon Cognito actions"""  
    def __init__(self, cognito_idp_client, user_pool_id, client_id,  
                 client_secret=None):  
        """  
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.  
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.  
        :param client_id: The ID of a client application registered with the user  
        pool.  
        :param client_secret: The client secret, if the client has a secret.  
        """  
        self.cognito_idp_client = cognito_idp_client  
        self.user_pool_id = user_pool_id  
        self.client_id = client_id  
        self.client_secret = client_secret  
  
    def confirm_user_sign_up(self, user_name, confirmation_code):  
        """  
        Confirms a previously created user. A user must be confirmed before they  
        can sign in to Amazon Cognito.  
  
        :param user_name: The name of the user to confirm.  
        :param confirmation_code: The confirmation code sent to the user's  
        registered  
                           email address.  
        :return: True when the confirmation succeeds.  
        """  
        try:  
            kwargs = {
```

```
'ClientId': self.client_id, 'Username': user_name,
'ConfirmationCode': confirmation_code}
if self.client_secret is not None:
    kwargs['SecretHash'] = self._secret_hash(user_name)
self.cognito_idp_client.confirm_sign_up(**kwargs)
except ClientError as err:
    logger.error(
        "Couldn't confirm sign up for %s. Here's why: %s: %s",
        user_name,
        err.response['Error']['Code'],
        err.response['Error']['Message'])
    raise
else:
    return True
```

- For API details, see [ConfirmSignUp](#) in *AWS SDK for Python (Boto3) API Reference*.

## Confirm an MFA device for tracking by Amazon Cognito using an AWS SDK

The following code examples show how to confirm an MFA device for tracking by Amazon Cognito.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const confirmDevice = async ({ deviceKey, accessToken, passwordVerifier, salt }) =>
{
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new ConfirmDeviceCommand({
    DeviceKey: deviceKey,
    AccessToken: accessToken,
    DeviceSecretVerifierConfig: {
      PasswordVerifier: passwordVerifier,
      Salt: salt
    },
  });

  return client.send(command);
};
```

- For API details, see [ConfirmDevice](#) in *AWS SDK for JavaScript API Reference*.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:
```

```

"""Encapsulates Amazon Cognito actions"""
def __init__(self, cognito_idp_client, user_pool_id, client_id,
client_secret=None):
    """
    :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
    :param user_pool_id: The ID of an existing Amazon Cognito user pool.
    :param client_id: The ID of a client application registered with the user
pool.
    :param client_secret: The client secret, if the client has a secret.
    """
    self.cognito_idp_client = cognito_idp_client
    self.user_pool_id = user_pool_id
    self.client_id = client_id
    self.client_secret = client_secret

    def confirm_mfa_device(
        self, user_name, device_key, device_group_key, device_password,
access_token,
        aws_srp):
    """
    Confirms an MFA device to be tracked by Amazon Cognito. When a device is
tracked, its key and password can be used to sign in without requiring a
new
MFA code from the MFA application.

    :param user_name: The user that is associated with the device.
    :param device_key: The key of the device, returned by Amazon Cognito.
    :param device_group_key: The group key of the device, returned by Amazon
Cognito.
    :param device_password: The password that is associated with the device.
    :param access_token: The user's access token.
    :param aws_srp: A class that helps with Secure Remote Password (SRP)
calculations. The scenario associated with this example
uses
            the warrant package.
    :return: True when the user must confirm the device. Otherwise, False. When
        False, the device is automatically confirmed and tracked.
    """
    srp_helper = aws_srp.AWSSRP(
        username=user_name, password=device_password,
        pool_id='__', client_id=self.client_id, client_secret=None,
        client=self.cognito_idp_client)
    device_and_pw = f'{device_group_key}{device_key}:{device_password}'
    device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode('utf-8'))
    salt = aws_srp.pad_hex(aws_srp.get_random(16))
    x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt + device_and_pw_hash))
    verifier = aws_srp.pad_hex(pow(srp_helper.g, x_value, srp_helper.big_n))
    device_secret_verifier_config = {
        "PasswordVerifier":
base64.standard_b64encode(bytearray.fromhex(verifier)).decode('utf-8'),
        "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode('utf-8')
    }
    try:
        response = self.cognito_idp_client.confirm_device(
            AccessToken=access_token,
            DeviceKey=device_key,
            DeviceSecretVerifierConfig=device_secret_verifier_config)
        user_confirm = response['UserConfirmationNecessary']
    except ClientError as err:
        logger.error(
            "Couldn't confirm mfa device %s. Here's why: %s: %s",
            device_key,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return user_confirm

```

- For API details, see [ConfirmDevice](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a token to associate an MFA application with an Amazon Cognito user using an AWS SDK

The following code examples show how to get a token to associate an MFA application with an Amazon Cognito user.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Gets the secret token that will enable multi-factor
/// authentication (MFA) for the user.
/// </summary>
/// <param name="identityProviderClient">An initialized Identity
/// Provider client object.</param>
/// <param name="session">The currently active session.</param>
/// <returns>Returns a string representing the currently active
/// session.</returns>
public static async Task<string> GetSecretForAppMFA(
    AmazonCognitoIdentityProviderClient identityProviderClient,
    string session)
{
    var softwareTokenRequest = new AssociateSoftwareTokenRequest
    {
        Session = session,
    };

    var tokenResponse = await
identityProviderClient.AssociateSoftwareTokenAsync(softwareTokenRequest);
    var secretCode = tokenResponse.SecretCode;

    Console.WriteLine($"Enter the following token into Google
Authenticator: {secretCode}");

    return tokenResponse.Session;
}
```

- For API details, see [AssociateSoftwareToken](#) in *AWS SDK for .NET API Reference*.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {

    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
        .session(session)
        .build();

    AssociateSoftwareTokenResponse tokenResponse =
identityProviderClient.associateSoftwareToken(softwareTokenRequest) ;
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}
```

- For API details, see [AssociateSoftwareToken](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const associateSoftwareToken = async (session) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);
  const command = new AssociateSoftwareTokenCommand({
    Session: session,
  });

  return client.send(command);
};
```

- For API details, see [AssociateSoftwareToken](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {

    val softwareTokenRequest = AssociateSoftwareTokenRequest {
        session = sessionVal
    }
```

```
CognitoIdentityProviderClient { region = "us-east-1" }.use
{ identityProviderClient ->
    val tokenResponse =
        identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
}
```

- For API details, see [AssociateSoftwareToken](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
                          pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret

    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the
        user.

        :param session: Session information returned from a previous call to
                        initiate
                           authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
                self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            response.pop('ResponseMetadata', None)
        return response
```

- For API details, see [AssociateSoftwareToken](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get information about an Amazon Cognito user using an AWS SDK

The following code examples show how to get information about an Amazon Cognito user.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Checks the status of a user for a particular Amazon Cognito user
/// pool.
/// </summary>
/// <param name="identityProviderClient">An initialized Identity
/// Provider client object.</param>
/// <param name="userName">The user name for which we want to check
/// the status.</param>
/// <param name="poolId">The user pool for which we want to check the
/// user's status.</param>
/// <returns>The UserStatusType object indicating the user's status.</returns>
public static async Task<UserStatusType>
GetAdminUser(AmazonCognitoIdentityProviderClient identityProviderClient, string
userName, string poolId)
{
    var userRequest = new Admin GetUserRequest
    {
        Username = userName,
        UserPoolId = poolId,
    };

    var response = await
identityProviderClient.Admin GetUserAsync(userRequest);

    Console.WriteLine($"User status {response.UserStatus}");
    return response.UserStatus;
}
```

- For API details, see [Admin GetUser](#) in *AWS SDK for .NET API Reference*.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName, String poolId) {
```

```
try {
    Admin GetUserRequest userRequest = Admin GetUserRequest.builder()
        .username(userName)
        .userPoolId(poolId)
        .build();

    Admin GetUserResponse response =
    identityProviderClient.admin GetUser(userRequest);
    System.out.println("User status "+response.userStatusAsString());

} catch (CognitoIdentityProviderException e){
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [Admin GetUser](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const admin GetUser = async ({ userPoolId, username }) => {
    const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

    const command = new Admin GetUserCommand({
        UserPoolId: userPoolId,
        Username: username,
    });

    return client.send(command);
};
```

- For API details, see [Admin GetUser](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getAdminUser(userNameVal: String?, poolIdVal: String?) {

    val userRequest = Admin GetUserRequest {
        username = userNameVal
        userPoolId = poolIdVal
    }
}
```

```
        }

        CognitoIdentityProviderClient { region = "us-east-1" }.use
{ identityProviderClient ->
    val response = identityProviderClient.adminGetUser(userRequest)
    println("User status ${response.userStatus}")
}
}
```

- For API details, see [Admin GetUser](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret

    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon Cognito
        to send an email to the specified email address. The email contains a code
        that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
        whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                Otherwise, false.
        """
        try:
            kwargs = {
                'ClientId': self.client_id, 'Username': user_name, 'Password':
password,
                'UserAttributes': [{'Name': 'email', 'Value': user_email}]}
            if self.client_secret is not None:
                kwargs['SecretHash'] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response['UserConfirmed']
        except ClientError as err:
```

```
        if err.response['Error']['Code'] == 'UsernameExistsException':
            response = self.cognito_idp_client.admin_get_user(
                UserPoolId=self.user_pool_id, Username=user_name)
            logger.warning("User %s exists and is %s.", user_name,
                           response['UserStatus'])
            confirmed = response['UserStatus'] == 'CONFIRMED'
        else:
            logger.error(
                "Couldn't sign up %s. Here's why: %s: %s",
                user_name, err.response['Error']['Code'], err.response['Error']
                ['Message'])
            raise
    return confirmed
```

- For API details, see [Admin GetUser](#) in *AWS SDK for Python (Boto3) API Reference*.

## List the Amazon Cognito user pools using an AWS SDK

The following code example shows how to list the Amazon Cognito user pools.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client.list_user_pools().max_results(10).send().await?;
    if let Some(pools) = response.user_pools() {
        println!("User pools:");
        for pool in pools {
            println!("  ID:          {}", pool.id().unwrap_or_default());
            println!("  Name:         {}", pool.name().unwrap_or_default());
            println!("  Status:       {:?}", pool.status());
            println!("  Lambda Config: {:?}", pool.lambda_config().unwrap());
            println!(
                "  Last modified:  {}",
                pool.last_modified_date().unwrap().to_chrono_utc()
            );
            println!(
                "  Creation date:  {}",
                pool.creation_date().unwrap().to_chrono_utc()
            );
            println!();
        }
    }
    println!("Next token: {}", response.next_token().unwrap_or_default());
    Ok(())
}
```

- For API details, see [ListUserPools](#) in *AWS SDK for Rust API reference*.

## List Amazon Cognito users using an AWS SDK

The following code examples show how to list Amazon Cognito users.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId ) {
    try {
        ListUsersRequest usersRequest = ListUsersRequest.builder()
            .userPoolId(userPoolId)
            .build();

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created " + user.userCreateDate() );
        });
    } catch (CognitoIdentityProviderException e){
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Shows how to list users by using a filter.
public static void listUsersFilter(CognitoIdentityProviderClient cognitoClient,
String userPoolId ) {

    try {
        String filter = "email = \"tblue@noserver.com\"";
        ListUsersRequest usersRequest = ListUsersRequest.builder()
            .userPoolId(userPoolId)
            .filter(filter)
            .build();

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User with filter applied " + user.username() +
" Status " + user.userStatus() + " Created " + user.userCreateDate() );
        });
    } catch (CognitoIdentityProviderException e){
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const listUsers = async ({ userPoolId }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new ListUsersCommand({
    UserPoolId: userPoolId,
  });

  return client.send(command);
};
```

- For API details, see [ListUsers](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAllUsers(userPoolId: String) {

    val request = ListUsersRequest {
        this.userPoolId = userPoolId
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:
```

```
"""Encapsulates Amazon Cognito actions"""
def __init__(self, cognito_idp_client, user_pool_id, client_id,
client_secret=None):
    """
    :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
    :param user_pool_id: The ID of an existing Amazon Cognito user pool.
    :param client_id: The ID of a client application registered with the user
pool.
    :param client_secret: The client secret, if the client has a secret.
    """
    self.cognito_idp_client = cognito_idp_client
    self.user_pool_id = user_pool_id
    self.client_id = client_id
    self.client_secret = client_secret

def list_users(self):
    """
    Returns a list of the users in the current user pool.

    :return: The list of users.
    """
    try:
        response =
self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
        users = response['Users']
    except ClientError as err:
        logger.error(
            "Couldn't list users for %s. Here's why: %s: %s",
            self.user_pool_id,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return users
```

- For API details, see [ListUsers](#) in *AWS SDK for Python (Boto3) API Reference*.

## Resend an Amazon Cognito confirmation code using an AWS SDK

The following code examples show how to resend an Amazon Cognito confirmation code.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Causes the confirmation code for user registration to be sent
/// again.
/// </summary>
/// <param name="identityProviderClient">An initialized Identity
/// Provider client object.</param>
/// <param name="clientId">The client Id of the application associated
/// with the user pool.</param>
/// <param name="userName">The user name to be confirmed.</param>
/// <returns>A System Threading Task.</returns>
```

```
    public static async Task<Unit> ResendConfirmationCode(AmazonCognitoIdentityProviderClient identityProviderClient, string clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
identityProviderClient.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
{response.CodeDeliveryDetails.DeliveryMedium}");
    }
}
```

- For API details, see [ResendConfirmationCode](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void resendConfirmationCode(CognitoIdentityProviderClient identityProviderClient, String clientId, String userName) {

    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
        .clientId(clientId)
        .username(userName)
        .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is
"+response.codeDeliveryDetails().deliveryMediumAsString());

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ResendConfirmationCode](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const resendConfirmationCode = async ({ clientId, username }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new ResendConfirmationCodeCommand({
    ClientId: clientId,
    Username: username
  });

  return client.send(command);
};
```

- For API details, see [ResendConfirmationCode](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun resendConfirmationCode(clientIdVal: String?, userNameVal: String?) {

    val codeRequest = ResendConfirmationCodeRequest {
        clientId = clientIdVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
            (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- For API details, see [ResendConfirmationCode](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        """


```

```
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
"""
self.cognito_idp_client = cognito_idp_client
self.user_pool_id = user_pool_id
self.client_id = client_id
self.client_secret = client_secret

def resend_confirmation(self, user_name):
    """
    Prompts Amazon Cognito to resend an email with a new confirmation code.

    :param user_name: The name of the user who will receive the email.
    :return: Delivery information about where the email is sent.
    """
    try:
        kwargs = {
            'ClientId': self.client_id, 'Username': user_name}
        if self.client_secret is not None:
            kwargs['SecretHash'] = self._secret_hash(user_name)
        response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
        delivery = response['CodeDeliveryDetails']
    except ClientError as err:
        logger.error(
            "Couldn't resend confirmation to %s. Here's why: %s: %s",
            user_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return delivery
```

- For API details, see [ResendConfirmationCode](#) in *AWS SDK for Python (Boto3) API Reference*.

## Respond to Amazon Cognito SRP authentication challenges using an AWS SDK

The following code examples show how to respond to Amazon Cognito SRP authentication challenges.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Responds to an authentication challenge for an Amazon Cognito user.
/// </summary>
/// <param name="identityProviderClient">The Amazon Cognito client
object.</param>
/// <param name="userName">The user name of the user to authenticate.</
param>
/// <param name="clientId">The client Id of the application associated
/// with the user pool.</param>
/// <param name="mfaCode">The MFA code supplied by the user.</param>
/// <param name="session">The session for which the user will be
authenticated.</param>
```

```
    /// <returns>A Boolean value that indicates the success of the authentication.</returns>
    public static async Task<bool> AdminRespondToAuthChallenge(
        AmazonCognitoIdentityProviderClient identityProviderClient,
        string userName,
        string clientId,
        string mfaCode,
        string session)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>
        {
            { "USERNAME", userName },
            { "SOFTWARE_TOKEN_MFA_CODE", mfaCode },
        };

        var respondToAuthChallengeRequest = new RespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
        };

        var response = await
identityProviderClient.RespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"response.AuthenticationResult()
{response.AuthenticationResult}");

        return response.AuthenticationResult is not null;
    }
}
```

- For API details, see [RespondToAuthChallenge](#) in [AWS SDK for .NET API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const respondToAuthChallenge = async ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
```

```
});  
  
    return client.send(command);  
};
```

- For API details, see [RespondToAuthChallenge](#) in *AWS SDK for JavaScript API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Sign in with a tracked device. To complete sign-in, the client must respond correctly to Secure Remote Password (SRP) challenges.

```
class CognitoIdentityProviderWrapper:  
    """Encapsulates Amazon Cognito actions"""  
    def __init__(self, cognito_idp_client, user_pool_id, client_id,  
                 client_secret=None):  
        """  
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.  
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.  
        :param client_id: The ID of a client application registered with the user  
        pool.  
        :param client_secret: The client secret, if the client has a secret.  
        """  
        self.cognito_idp_client = cognito_idp_client  
        self.user_pool_id = user_pool_id  
        self.client_id = client_id  
        self.client_secret = client_secret  
  
    def sign_in_with_tracked_device(  
            self, user_name, password, device_key, device_group_key,  
            device_password,  
            aws_srp):  
        """  
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in  
        with a tracked device lets a user sign in without entering a new MFA code.  
  
        Signing in with a tracked device requires that the client respond to the  
        SRP protocol. The scenario associated with this example uses the warrant  
        package  
        to help with SRP calculations.  
  
        For more information on SRP, see https://en.wikipedia.org/wiki/Secure\_Remote\_Password\_protocol.  
  
        :param user_name: The user that is associated with the device.  
        :param password: The user's password.  
        :param device_key: The key of a tracked device.  
        :param device_group_key: The group key of a tracked device.  
        :param device_password: The password that is associated with the device.  
        :param aws_srp: A class that helps with SRP calculations. The scenario  
                      associated with this example uses the warrant package.  
        :return: The result of the authentication. When successful, this contains  
                an  
                access token for the user.  
        """
```

```
try:
    srp_helper = aws_srp.AWSSRP(
        username=user_name, password=device_password,
        pool_id='_', client_id=self.client_id, client_secret=None,
        client=self.cognito_idp_client)

    response_init = self.cognito_idp_client.initiate_auth(
        ClientId=self.client_id, AuthFlow='USER_PASSWORD_AUTH',
        AuthParameters={
            'USERNAME': user_name, 'PASSWORD': password, 'DEVICE_KEY':
device_key})
    if response_init['ChallengeName'] != 'DEVICE_SRP_AUTH':
        raise RuntimeError(
            f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}.")

    auth_params = srp_helper.get_auth_params()
    auth_params['DEVICE_KEY'] = device_key
    response_auth = self.cognito_idp_client.respond_to_auth_challenge(
        ClientId=self.client_id,
        ChallengeName='DEVICE_SRP_AUTH',
        ChallengeResponses=auth_params
    )
    if response_auth['ChallengeName'] != 'DEVICE_PASSWORD_VERIFIER':
        raise RuntimeError(
            f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
            f"{response_init['ChallengeName']}.")

    challenge_params = response_auth['ChallengeParameters']
    challenge_params['USER_ID_FOR_SRP'] = device_group_key + device_key
    cr = srp_helper.process_challenge(challenge_params)
    cr['USERNAME'] = user_name
    cr['DEVICE_KEY'] = device_key
    response_verifier = self.cognito_idp_client.respond_to_auth_challenge(
        ClientId=self.client_id,
        ChallengeName='DEVICE_PASSWORD_VERIFIER',
        ChallengeResponses=cr)
    auth_tokens = response_verifier['AuthenticationResult']
except ClientError as err:
    logger.error(
        "Couldn't start client sign in for %s. Here's why: %s: %s",
        user_name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return auth_tokens
```

- For API details, see [RespondToAuthChallenge](#) in *AWS SDK for Python (Boto3) API Reference*.

## Respond to an Amazon Cognito authentication challenge using an AWS SDK

The following code examples show how to respond to an Amazon Cognito authentication challenge.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    ///<summary>
    /// Responds to an authentication challenge for an Amazon Cognito user.
    ///</summary>
    ///<param name="identityProviderClient">The Amazon Cognito client
    object.</param>
    ///<param name="userName">The user name of the user to authenticate.</param>
    ///<param name="clientId">The client Id of the application associated
    ///with the user pool.</param>
    ///<param name="mfaCode">The MFA code supplied by the user.</param>
    ///<param name="session">The session for which the user will be
    authenticated.</param>
    ///<returns>A Boolean value that indicates the success of the
    authentication.</returns>
    public static async Task<bool> AdminRespondToAuthChallenge(
        AmazonCognitoIdentityProviderClient identityProviderClient,
        string userName,
        string clientId,
        string mfaCode,
        string session)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>
        {
            { "USERNAME", userName },
            { "SOFTWARE_TOKEN_MFA_CODE", mfaCode },
        };

        var respondToAuthChallengeRequest = new RespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
        };

        var response = await
identityProviderClient.RespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"response.getAuthenticationResult()
{response.AuthenticationResult}");

        return response.AuthenticationResult is not null;
    }
}
```

- For API details, see [AdminRespondToAuthChallenge](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Respond to an authentication challenge.
public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient, String userName, String clientId, String mfaCode, String
session) {
```

```
System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
Map<String, String> challengeResponses = new HashMap<>();

challengeResponses.put("USERNAME", userName);
challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

RespondToAuthChallengeRequest respondToAuthChallengeRequest =
RespondToAuthChallengeRequest.builder()
    .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
    .clientId(clientId)
    .challengeResponses(challengeResponses)
    .session(session)
    .build();

RespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient.respondToAuthChallenge(respondToAuthChallengeRequest);
System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
+ respondToAuthChallengeResult.authenticationResult());
}
```

- For API details, see [AdminRespondToAuthChallenge](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const adminRespondToAuthChallenge = async ({
  userPoolId,
  clientId,
  username,
  totp,
  session,
}) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);
  const command = new AdminRespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: totp,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- For API details, see [AdminRespondToAuthChallenge](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(userName: String, clientIdVal: String?, mfaCode: String, sessionVal: String?) {

    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponses0b = mutableMapOf<String, String>()
    challengeResponses0b["USERNAME"] = userName
    challengeResponses0b["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val respondToAuthChallengeRequest = RespondToAuthChallengeRequest {
        challengeName = ChallengeNameType.SoftwareTokenMfa
        clientId = clientIdVal
        challengeResponses = challengeResponses0b
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val respondToAuthChallengeResult =
        identityProviderClient.respondToAuthChallenge(respondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()\n${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- For API details, see [AdminRespondToAuthChallenge](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Respond to an MFA challenge by providing a code generated by an associated MFA application.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
        pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
```

```
        self.client_id = client_id
        self.client_secret = client_secret

    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step of
        a two-factor sign-in. When sign-in is successful, it returns an access
        token
            that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
        initiate
                    authentication.
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains
        an
                    access token for the user.
        """
        try:
            kwargs = {
                'UserPoolId': self.user_pool_id,
                'ClientId': self.client_id,
                'ChallengeName': 'SOFTWARE_TOKEN_MFA', 'Session': session,
                'ChallengeResponses': {
                    'USERNAME': user_name, 'SOFTWARE_TOKEN_MFA_CODE': mfa_code}}
            if self.client_secret is not None:
                kwargs['ChallengeResponses']['SECRET_HASH'] =
self._secret_hash(user_name)
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response['AuthenticationResult']
            except ClientError as err:
                if err.response['Error']['Code'] == 'ExpiredCodeException':
                    logger.warning(
                        "Your MFA code has expired or has been used already. You might
have "
                        "to wait a few seconds until your app shows you a new code.")
                else:
                    logger.error(
                        "Couldn't respond to mfa challenge for %s. Here's why: %s: %s",
user_name,
                        err.response['Error']['Code'], err.response['Error']
['Message'])
                    raise
            else:
                return auth_result
        
```

- For API details, see [AdminRespondToAuthChallenge in AWS SDK for Python \(Boto3\) API Reference](#).

## Sign up a user with Amazon Cognito using an AWS SDK

The following code examples show how to sign up a user with Amazon Cognito.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Add a new user to an Amazon Cognito user pool.
/// </summary>
/// <param name="identityProviderClient">An initialized Identity
/// Provider client object.</param>
/// <param name="clientId">The client Id of the application associated
/// with the user pool.</param>
/// <param name="userName">The user name of the user to sign up.</param>
/// <param name="password">The password for the user.</param>
/// <param name="email">The user's email address.</param>
/// <returns>A System Threading Task.</returns>
public static async Task<string> SignUp(
    AmazonCognitoIdentityProviderClient identityProviderClient,
    string clientId,
    string userName,
    string password,
    string email)
{
    var userAttrs = new AttributeType
    {
        Name = "email",
        Value = email,
    };

    var userAttrsList = new List<AttributeType>
    {
        userAttrs,
    };

    var signUpRequest = new SignUpRequest
    {
        UserAttributes = userAttrsList,
        Username = userName,
        ClientId = clientId,
        Password = password,
    };

    var response = await identityProviderClient.SignUpAsync(signUpRequest);
    Console.WriteLine("User has been signed up.");
    return response.UserSub;
}
```

- For API details, see [SignUp](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName, String password, String email) {

    AttributeType userAttrs = AttributeType.builder()
        .name("email")
        .value(email)
```

```
.build();

List<AttributeType> userAttrsList = new ArrayList<>();
userAttrsList.add(userAttrs);

try {
    SignUpRequest signUpRequest = SignUpRequest.builder()
        .userAttributes(userAttrsList)
        .username(userName)
        .clientId(clientId)
        .password(password)
        .build();

    identityProviderClient.signUp(signUpRequest);
    System.out.println("User has been signed up");

} catch(CognitoIdentityProviderException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [SignUp](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const signUp = async ({ clientId, username, password, email }) => {
    const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

    const command = new SignUpCommand({
        ClientId: clientId,
        Username: username,
        Password: password,
        UserAttributes: [{ Name: "email", Value: email }],
    });

    return client.send(command);
};
```

- For API details, see [SignUp](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun signUp(clientIdVal: String?, userNameVal: String?, passwordVal: String?, emailVal: String?) {

    val userAttrs = AttributeType {
        name = "email"
        value = emailVal
    }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)

    val signUpRequest = SignUpRequest {
        userAttributes = userAttrsList
        username = userNameVal
        clientId = clientIdVal
        password = passwordVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- For API details, see [SignUp](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
        pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret

    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon Cognito
        to send an email to the specified email address. The email contains a code
        that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
        whether

```

```
the user has been confirmed.

:param user_name: The user name that identifies the new user.
:param password: The password for the new user.
:param user_email: The email address for the new user.
:return: True when the user is already confirmed with Amazon Cognito.
        Otherwise, false.
"""
try:
    kwargs = {
        'ClientId': self.client_id, 'Username': user_name, 'Password': password,
        'UserAttributes': [{'Name': 'email', 'Value': user_email}]}
    if self.client_secret is not None:
        kwargs['SecretHash'] = self._secret_hash(user_name)
    response = self.cognito_idp_client.sign_up(**kwargs)
    confirmed = response['UserConfirmed']
except ClientError as err:
    if err.response['Error']['Code'] == 'UsernameExistsException':
        response = self.cognito_idp_client.admin_get_user(
            UserPoolId=self.user_pool_id, Username=user_name)
        logger.warning("User %s exists and is %s.", user_name,
                       response['UserStatus'])
        confirmed = response['UserStatus'] == 'CONFIRMED'
    else:
        logger.error(
            "Couldn't sign up %s. Here's why: %s: %s",
            user_name, err.response['Error']['Code'], err.response['Error']
            ['Message'])
        raise
return confirmed
```

- For API details, see [SignUp](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start authentication with a device tracked by Amazon Cognito using an AWS SDK

The following code examples show how to start authentication with a device tracked by Amazon Cognito.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Initiates the authorization process.
/// </summary>
/// <param name="identityProviderClient">An initialized Identity
/// Provider client object.</param>
/// <param name="clientId">The client Id of the application associated
/// with the user pool.</param>
/// <param name="userName">The user name to be authorized.</param>
/// <param name="password">The password of the user.</param>
/// <returns>The response from the client from the InitiateAuthAsync
/// call.</returns>
```

```
    public static async Task<InitiateAuthResponse>
InitiateAuth(AmazonCognitoIdentityProviderClient identityProviderClient, string
clientId, string userName, string password)
{
    var authParameters = new Dictionary<string, string>
    {
        { "USERNAME", userName },
        { "PASSWORD", password },
    };

    var authRequest = new InitiateAuthRequest
    {
        ClientId = clientId,
        AuthParameters = authParameters,
        AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
    };

    var response = await
identityProviderClient.InitiateAuthAsync(authRequest);
    Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

    return response;
}
```

- For API details, see [InitiateAuth in AWS SDK for .NET API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const initiateAuth = async ({ username, password, clientId }) => {
    const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

    const command = new InitiateAuthCommand({
        AuthFlow: AuthFlowType.USER_PASSWORD_AUTH,
        AuthParameters: {
            USERNAME: username,
            PASSWORD: password,
        },
        ClientId: clientId,
    });

    return client.send(command);
};
```

- For API details, see [InitiateAuth in AWS SDK for JavaScript API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Sign in with a tracked device. To complete sign-in, the client must respond correctly to Secure Remote Password (SRP) challenges.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
        pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret

    def sign_in_with_tracked_device(
            self, user_name, password, device_key, device_group_key,
            device_password,
            aws_srp):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA code.

        Signing in with a tracked device requires that the client respond to the
        SRP
        protocol. The scenario associated with this example uses the warrant
        package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/Secure\_Remote\_Password\_protocol.
        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
        an
                access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name, password=device_password,
                pool_id='__', client_id=self.client_id, client_secret=None,
                client=self.cognito_idp_client)

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id, AuthFlow='USER_PASSWORD_AUTH',
                AuthParameters={
                    'USERNAME': user_name, 'PASSWORD': password, 'DEVICE_KEY':
device_key})
            if response_init['ChallengeName'] != 'DEVICE_SRP_AUTH':
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}.")

            auth_params = srp_helper.get_auth_params()
            auth_params['DEVICE_KEY'] = device_key
        except Exception as e:
            print(f"Error during sign-in: {e}")
            return None
        return response_init
```

```
        response_auth = self.cognito_idp_client.respond_to_auth_challenge(
            ClientId=self.client_id,
            ChallengeName='DEVICE_SRP_AUTH',
            ChallengeResponses=auth_params
        )
        if response_auth['ChallengeName'] != 'DEVICE_PASSWORD_VERIFIER':
            raise RuntimeError(
                f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                f"{response_init['ChallengeName']}")

        challenge_params = response_auth['ChallengeParameters']
        challenge_params['USER_ID_FOR_SRP'] = device_group_key + device_key
        cr = srp_helper.process_challenge(challenge_params)
        cr['USERNAME'] = user_name
        cr['DEVICE_KEY'] = device_key
        response_verifier = self.cognito_idp_client.respond_to_auth_challenge(
            ClientId=self.client_id,
            ChallengeName='DEVICE_PASSWORD_VERIFIER',
            ChallengeResponses=cr)
        auth_tokens = response_verifier['AuthenticationResult']
    except ClientError as err:
        logger.error(
            "Couldn't start client sign in for %s. Here's why: %s: %s",
            user_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return auth_tokens
```

- For API details, see [InitiateAuth in AWS SDK for Python \(Boto3\) API Reference](#).

## Start authentication with Amazon Cognito and administrator credentials using an AWS SDK

The following code examples show how to start authentication with Amazon Cognito and administrator credentials.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static InitiateAuthResponse initiateAuth(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName, String password) {

    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        InitiateAuthRequest authRequest = InitiateAuthRequest.builder()
            .clientId(clientId)
            .authParameters(authParameters)
            .authFlow(AuthFlowType.USER_PASSWORD_AUTH)
            .build();
```

```
        InitiateAuthResponse response =
    identityProviderClient.initiateAuth(authRequest);
    System.out.println("Result Challenge is : " +
response.challengeName() );
    return response;

} catch(CognitoIdentityProviderException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return null;
}
```

- For API details, see [AdminInitiateAuth](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const adminInitiateAuth = async ({
  clientId,
  userPoolId,
  username,
  password,
}) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};
```

- For API details, see [AdminInitiateAuth](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun initiateAuth(clientIdVal: String?, userNameVal: String, passwordVal:
String): InitiateAuthResponse {
```

```
val authParas = mutableMapOf <String, String>()
authParas["USERNAME"] = userNameVal
authParas["PASSWORD"] = passwordVal

val authRequest = InitiateAuthRequest {
    clientId = clientIdVal
    authParameters = authParas
    authFlow = AuthFlowType.UserPasswordAuth
}

CognitoIdentityProviderClient { region = "us-east-1" }.use
{ identityProviderClient ->
    val response = identityProviderClient.initiateAuth(authRequest)
    println("Result Challenge is ${response.challengeName}")
    return response
}
```

- For API details, see [AdminInitiateAuth](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
        pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret

    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
        server.

        If the user pool is configured to require MFA and this is the first sign-in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
        this
               returns an access token that can be used to get AWS credentials.
        Otherwise,
```

```
Amazon Cognito returns a challenge to set up an MFA application,
or a challenge to enter an MFA code from a registered MFA
application.
"""
try:
    kwargs = {
        'UserPoolId': self.user_pool_id,
        'ClientId': self.client_id,
        'AuthFlow': 'ADMIN_USER_PASSWORD_AUTH',
        'AuthParameters': {'USERNAME': user_name, 'PASSWORD': password}}
    if self.client_secret is not None:
        kwargs['AuthParameters']['SECRET_HASH'] =
self._secret_hash(user_name)
    response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
    challenge_name = response.get('ChallengeName', None)
    if challenge_name == 'MFA_SETUP':
        if 'SOFTWARE_TOKEN_MFA' in response['ChallengeParameters']
['MFAS_CAN_SETUP']:
            response.update(self.get_mfa_secret(response['Session']))
        else:
            raise RuntimeError(
                "The user pool requires MFA setup, but the user pool is not"
                "configured for TOTP MFA. This example requires TOTP MFA.")
    except ClientError as err:
        logger.error(
            "Couldn't start sign in for %s. Here's why: %s: %s",
            user_name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        response.pop('ResponseMetadata', None)
return response
```

- For API details, see [AdminInitiateAuth](#) in *AWS SDK for Python (Boto3) API Reference*.

## Verify an MFA application with an Amazon Cognito user using an AWS SDK

The following code examples show how to verify an MFA application with an Amazon Cognito user.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Verifies that the user has supplied the correct one-time password
/// and registers for multi-factor authentication (MFA).
/// </summary>
/// <param name="identityProviderClient">The Amazon Cognito client
object.</param>
/// <param name="session">The session for which the user will be
/// authenticated.</param>
/// <param name="code">The code provided by the user.</param>
/// <returns>A Boolean value that indicates the success of the
authentication.</returns>
```

```
public static async Task<bool> VerifyTOTP(
    AmazonCognitoIdentityProviderClient identityProviderClient,
    string session,
    string code)
{
    var tokenRequest = new VerifySoftwareTokenRequest
    {
        UserCode = code,
        Session = session,
    };

    var response = await
identityProviderClient.VerifySoftwareTokenAsync(tokenRequest);

    Console.WriteLine($"The status of the token is {response.Status}");

    return response.Status == VerifySoftwareTokenType.SUCCESS;
}
```

- For API details, see [VerifySoftwareToken](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {

    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)
            .session(session)
            .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is "
+verifyResponse.statusAsString());

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [VerifySoftwareToken](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const verifySoftwareToken = async (totp) => {
    const client = createClientForDefaultRegion(CognitoIdentityProviderClient);
    // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
    const session = process.env.SESSION;

    if (!session) {
        throw new Error(
            "Missing a valid Session. Did you run 'admin-initiate-auth'?"
        );
    }

    const command = new VerifySoftwareTokenCommand({
        Session: session,
        UserCode: totp,
    });

    return client.send(command);
};
```

- For API details, see [VerifySoftwareToken](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(sessionVal: String?, codeVal: String?) {

    val tokenRequest = VerifySoftwareTokenRequest {
        userCode = codeVal
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val verifyResponse =
            identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- For API details, see [VerifySoftwareToken](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
        pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret

    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
        initiate
                           authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code)
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            response.pop('ResponseMetadata', None)
        return response
```

- For API details, see [VerifySoftwareToken](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios for Amazon Cognito Identity Provider using AWS SDKs

The following code examples show how to use Amazon Cognito Identity Provider with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Sign up a user with an Amazon Cognito user pool that requires MFA using an AWS SDK \(p. 254\)](#)

## Sign up a user with an Amazon Cognito user pool that requires MFA using an AWS SDK

The following code examples show how to:

- Sign up a user with a user name, password, and email address.
- Confirm the user from a code sent in email.
- Set up multi-factor authentication by associating an MFA application with the user.
- Sign in by using a password and an MFA code.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
global using Amazon;
global using Amazon.CognitoIdentityProvider;
global using Amazon.CognitoIdentityProvider.Model;
global using Cognito_MVP;

// Before running this AWS SDK for .NET (v3) code example, set up your development
// environment, including your credentials.
// For more information, see the following documentation:
// https://docs.aws.amazon.com/sdk-for-net/v3/developer-guide/net-dg-setup.html
// TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
// CDK)
// script provided in this GitHub repo at:
// resources/cdk/cognito_scenario_user_pool_with_mfa.
// This code example performs the following operations:
// 1. Invokes the signUp method to sign up a user.
// 2. Invokes the adminGetUser method to get the user's confirmation status.
// 3. Invokes the ResendConfirmationCode method if the user requested another code.
// 4. Invokes the confirmSignUp method.
// 5. Invokes the initiateAuth to sign in. This results in being prompted to
//    set up TOTP (time-based one-time password). (The response is "ChallengeName": "MFA_SETUP").
// 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.
//    This can be used with Google Authenticator.
// 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
//    MFA.
// 8. Invokes the AdminInitiateAuth to sign in again. This results in being
//    prompted to submit a TOTP (Response: "ChallengeName": "SOFTWARE_TOKEN_MFA").
// 9. Invokes the AdminRespondToAuthChallenge to get back a token.

// Set the following variables:
// clientId - Fill in the app client Id value from the AWS CDK script.
string clientId = "";

// poolId - Fill in the pool Id that you can get from the AWS CDK script.
string poolId = "";
var userName = string.Empty;
var password = string.Empty;
var email = string.Empty;

string sepBar = new string('-', 80);
```

```
var identityProviderClient = new AmazonCognitoIdentityProviderClient();

do
{
    Console.Write("Enter your user name: ");
    userName = Console.ReadLine();
}
while (userName == string.Empty);

Console.WriteLine($"User name: {userName}");

do
{
    Console.Write("Enter your password: ");
    password = Console.ReadLine();
}
while (password == string.Empty);
Console.WriteLine($"Signing up {userName}");

do
{
    Console.Write("Enter your email: ");
    email = Console.ReadLine();
} while (email == string.Empty);

await CognitoMethods.SignUp(identityProviderClient, clientId, userName, password,
    email);

Console.WriteLine(sepBar);
Console.WriteLine($"Getting {userName} status from the user pool");
await CognitoMethods.GetAdminUser(identityProviderClient, userName, poolId);

Console.WriteLine(sepBar);
Console.WriteLine($"Conformation code sent to {userName}. Would you like to send a
    new code? (Yes/No)");
var ans = Console.ReadLine();

if (ans.ToUpper() == "YES")
{
    await CognitoMethods.ResendConfirmationCode(identityProviderClient, clientId,
        userName);
    Console.WriteLine("Sending a new confirmation code");
}

Console.WriteLine(sepBar);
Console.WriteLine("*** Enter confirmation code that was emailed");
string code = Console.ReadLine();

await CognitoMethods.ConfirmSignUp(identityProviderClient, clientId, code,
    userName);

Console.WriteLine($"Rechecking the status of {userName} in the user pool");
await CognitoMethods.GetAdminUser(identityProviderClient, userName, poolId);

var authResponse = await CognitoMethods.InitiateAuth(identityProviderClient,
    clientId, userName, password);
var mySession = authResponse.Session;

var newSession = await CognitoMethods.GetSecretForAppMFA(identityProviderClient,
    mySession);

Console.WriteLine("Enter the 6-digit code displayed in Google Authenticator");
string myCode = Console.ReadLine();

// Verify the TOTP and register for MFA.
await CognitoMethods.VerifyTOTP(identityProviderClient, newSession, myCode);
```

```
Console.WriteLine("Enter the new 6-digit code displayed in Google Authenticator");
string mfaCode = Console.ReadLine();

Console.WriteLine(sepBar);
var authResponse1 = await CognitoMethods.InitiateAuth(identityProviderClient,
    clientId, userName, password);
var session2 = authResponse1.Session;
await CognitoMethods.AdminRespondToAuthChallenge(identityProviderClient, userName,
    clientId, mfaCode, session2);

Console.WriteLine("The Cognito MVP application has completed.");
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [Admin GetUser](#)
  - [Admin Initiate Auth](#)
  - [Admin Respond To Auth Challenge](#)
  - [Associate Software Token](#)
  - [Confirm Device](#)
  - [Confirm Sign Up](#)
  - [Initiate Auth](#)
  - [List Users](#)
  - [Resend Confirmation Code](#)
  - [Respond To Auth Challenge](#)
  - [Sign Up](#)
  - [Verify Software Token](#)

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/** 
 * Before running this Java V2 code example, set up your development environment,
including your credentials.
*
* For more information, see the following documentation:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* TIP: To set up the required user pool, run the AWS Cloud Development
Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
cognito_scenario_user_pool_with_mfa.
*
* This code example performs the following operations:
*
* 1. Invokes the signUp method to sign up a user.
* 2. Invokes the adminGetUser method to get the user's confirmation status.
* 3. Invokes the ResendConfirmationCode method if the user requested another code.
* 4. Invokes the confirmSignUp method.
* 5. Invokes the initiateAuth to sign in. This results in being prompted to
set up TOTP (time-based one-time password). (The response is "ChallengeName":
"MFA_SETUP").
```

```
* 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.  
This can be used with Google Authenticator.  
* 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for  
MFA.  
* 8. Invokes the AdminInitiateAuth to sign in again. This results in being  
prompted to submit a TOTP (Response: "ChallengeName": "SOFTWARE_TOKEN_MFA").  
* 9. Invokes the AdminRespondToAuthChallenge to get back a token.  
*/  
  
public class CognitoMVP {  
    public static final String DASHES = new String(new char[80]).replace("\0",  
    "-");  
    public static void main(String[] args) throws NoSuchAlgorithmException,  
    InvalidKeyException {  
  
        final String usage = "\n" +  
            "Usage:\n" +  
            "  <clientId> <poolId>\n\n" +  
            "Where:\n" +  
            "  clientId - The app client Id value that you can get from the AWS  
CDK script.\n\n" +  
            "  poolId - The pool Id that you can get from the AWS CDK script. \n  
\n" ;  
  
        if (args.length != 2) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String clientId = args[0];  
        String poolId = args[1];  
        CognitoIdentityProviderClient identityProviderClient =  
CognitoIdentityProviderClient.builder()  
            .region(Region.US_EAST_1)  
            .credentialsProvider(ProfileCredentialsProvider.create())  
            .build();  
  
        System.out.println(DASHES);  
        System.out.println("Welcome to the Amazon Cognito example scenario.");  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("*** Enter your user name");  
        Scanner in = new Scanner(System.in);  
        String userName = in.nextLine();  
  
        System.out.println("*** Enter your password");  
        String password = in.nextLine();  
  
        System.out.println("*** Enter your email");  
        String email = in.nextLine();  
  
        System.out.println("1. Signing up " + userName);  
        signUp(identityProviderClient, clientId, userName, password, email);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("2. Getting " + userName + " in the user pool");  
        getAdminUser(identityProviderClient, userName, poolId);  
  
        System.out.println(" *** Conformation code sent to " + userName + ". Would  
you like to send a new code? (Yes/No)");  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        String ans = in.nextLine();
```

```
if (ans.compareTo("Yes") == 0) {
    resendConfirmationCode(identityProviderClient, clientId, userName);
    System.out.println("3. Sending a new confirmation code");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Enter confirmation code that was emailed");
String code = in.nextLine();
confirmSignUp(identityProviderClient, clientId, code, userName);
System.out.println("Rechecking the status of " + userName + " in the user
pool");
getAdminUser(identityProviderClient, userName, poolId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Invokes the initiateAuth to sign in");
InitiateAuthResponse authResponse = initiateAuth(identityProviderClient,
clientId, userName, password) ;
String mySession = authResponse.session() ;
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Invokes the AssociateSoftwareToken method to
generate a TOTP key");
String newSession = getSecretForAppMFA(identityProviderClient, mySession);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
String myCode = in.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Verify the TOTP and register for MFA");
verifyTOTP(identityProviderClient, newSession, myCode);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
String mfaCode = in.nextLine();
InitiateAuthResponse authResponse1 = initiateAuth(identityProviderClient,
clientId, userName, password);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Invokes the AdminRespondToAuthChallenge");
String session2 = authResponse1.session();
adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("All Amazon Cognito operations were successfully
performed");
System.out.println(DASHES);
}

// Respond to an authentication challenge.
public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient, String userName, String clientId, String mfaCode, String
session) {
```

```
System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
Map<String, String> challengeResponses = new HashMap<>();

challengeResponses.put("USERNAME", userName);
challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

RespondToAuthChallengeRequest respondToAuthChallengeRequest =
RespondToAuthChallengeRequest.builder()
    .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
    .clientId(clientId)
    .challengeResponses(challengeResponses)
    .session(session)
    .build();

RespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient.respondToAuthChallenge(respondToAuthChallengeRequest);
System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
+ respondToAuthChallengeResult.authenticationResult());
}

// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {

    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
    .userCode(code)
    .session(session)
    .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is "
+verifyResponse.statusAsString());

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static InitiateAuthResponse initiateAuth(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName, String password) {

    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        InitiateAuthRequest authRequest = InitiateAuthRequest.builder()
            .clientId(clientId)
            .authParameters(authParameters)
            .authFlow(AuthFlowType.USER_PASSWORD_AUTH)
            .build();

        InitiateAuthResponse response =
identityProviderClient.initiateAuth(authRequest);
        System.out.println("Result Challenge is : " +
response.challengeName() );
        return response;

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return null;
    }

    public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {

        AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
        .session(session)
        .build();

        AssociateSoftwareTokenResponse tokenResponse =
identityProviderClient.associateSoftwareToken(softwareTokenRequest) ;
        String secretCode = tokenResponse.secretCode();
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }

    public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code, String userName) {

        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                .clientId(clientId)
                .confirmationCode(code)
                .username(userName)
                .build();

            identityProviderClient.confirmSignUp(signUpRequest);
            System.out.println(userName +" was confirmed");

        } catch(CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName) {

        try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is
"+response.codeDeliveryDetails().deliveryMediumAsString());

        } catch(CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName, String password, String email) {

        AttributeType userAttrs = AttributeType.builder()
            .name("email")
            .value(email)
```

```
.build();

List<AttributeType> userAttrsList = new ArrayList<>();
userAttrsList.add(userAttrs);

try {
    SignUpRequest signUpRequest = SignUpRequest.builder()
        .userAttributes(userAttrsList)
        .username(userName)
        .clientId(clientId)
        .password(password)
        .build();

    identityProviderClient.signUp(signUpRequest);
    System.out.println("User has been signed up");

} catch(CognitoIdentityProviderException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName, String poolId) {

    try {
        Admin GetUserRequest userRequest = Admin GetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        Admin GetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
        System.out.println("User status "+response.userStatusAsString());

    } catch (CognitoIdentityProviderException e){
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [Admin GetUser](#)
  - [Admin Initiate Auth](#)
  - [Admin Respond To Auth Challenge](#)
  - [Associate Software Token](#)
  - [Confirm Device](#)
  - [Confirm Sign Up](#)
  - [Initiate Auth](#)
  - [List Users](#)
  - [Resend Confirmation Code](#)
  - [Respond To Auth Challenge](#)
  - [Sign Up](#)
  - [Verify Software Token](#)

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

For the best experience, clone the GitHub repository and run this example. The following code represents a sample of the full example application.

```
import { log } from "../../../../../libs/utils/util-log.js";
import { signUp } from "../../../../../actions/sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "../../../../../libs/utils/util-csv.js";

const validateClient = (clientId) => {
    if (!clientId) {
        throw new Error(
            `App client id is missing. Did you run 'create-user-pool'?`
        );
    }
};

const validateUser = (username, password, email) => {
    if (!(username && password && email)) {
        throw new Error(
            `Username, password, and email must be provided as arguments to the 'sign-up' command.`
        );
    }
};

const signUpHandler = async (commands) => {
    const [_, username, password, email] = commands;

    try {
        validateUser(username, password, email);
        const clientId = getSecondValuesFromEntries(FILE_USER_POOLS)[0];
        validateClient(clientId);
        log(`Signing up.`);
        await signUp({ clientId, username, password, email });
        log(`Signed up. An confirmation email has been sent to: ${email}.`);
        log(`Run 'confirm-sign-up ${username} <code>' to confirm your account.`);
    } catch (err) {
        log(err);
    }
};

export { signUpHandler };

const signUp = async ({ clientId, username, password, email }) => {
    const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

    const command = new SignUpCommand({
        ClientId: clientId,
        Username: username,
        Password: password,
        UserAttributes: [{ Name: "email", Value: email }],
    });

    return client.send(command);
};

import { log } from "../../../../../libs/utils/util-log.js";
```

```
import { confirmSignUp } from "../../actions/confirm-sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "../../../../../libs/utils/util-csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`
    );
  }
};

const validateUser = (username) => {
  if (!username) {
    throw new Error(
      `Username name is missing. It must be provided as an argument to the 'confirm-sign-up' command.`
    );
  }
};

const validateCode = (code) => {
  if (!code) {
    throw new Error(
      `Verification code is missing. It must be provided as an argument to the 'confirm-sign-up' command.`
    );
  }
};

const confirmSignUpHandler = async (commands) => {
  const [_, username, code] = commands;

  try {
    validateUser(username);
    validateCode(code);
    const clientId = getSecondValuesFromEntries(FILE_USER_POOLS)[0];
    validateClient(clientId);
    log(`Confirming user.`);
    await confirmSignUp({ clientId, username, code });
    log(`User confirmed. Run 'admin-initiate-auth ${username} <password>' to sign in.`);
  } catch (err) {
    log(err);
  }
};

export { confirmSignUpHandler };

const confirmSignUp = async ({ clientId, username, code }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};

import qrcode from "qrcode-terminal";
import { log } from "../../../../../libs/utils/util-log.js";
import { adminInitiateAuth } from "../../../../../actions/admin-initiate-auth.js";
```

```
import { associateSoftwareToken } from "../../actions/associate-software-
token.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getFirstEntry } from "../../libs/utils/util-csv.js";

const handleMfaSetup = async (session, username) => {
    const { SecretCode, Session } = await associateSoftwareToken(session);

    // Store the Session for use with 'VerifySoftwareToken'.
    process.env.SESSION = Session;

    console.log(
        "Scan this code in your preferred authenticator app, then run 'verify-software-
        token' to finish the setup."
    );
    qrCode.generate(
        `otpauth://totp/${username}?secret=${SecretCode}`,
        { small: true },
        console.log
    );
};

const handleSoftwareTokenMfa = (session) => {
    // Store the Session for use with 'AdminRespondToAuthChallenge'.
    process.env.SESSION = session;
};

const validateClient = (id) => {
    if (!id) {
        throw new Error(
            `User pool client id is missing. Did you run 'create-user-pool?'`
        );
    }
};

const validateId = (id) => {
    if (!id) {
        throw new Error(`User pool id is missing. Did you run 'create-user-pool?'`);
    }
};

const validateUser = (username, password) => {
    if (!(username && password)) {
        throw new Error(
            `Username and password must be provided as arguments to the 'admin-initiate-
            auth' command.`
        );
    }
};

const adminInitiateAuthHandler = async (commands) => {
    const [_, username, password] = commands;

    try {
        validateUser(username, password);

        const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
        validateId(userPoolId);
        validateClient(clientId);

        log("Signing in.");
        const { ChallengeName, Session } = await adminInitiateAuth({
            clientId,
            userPoolId,
            username,
            password,
        });
    }
};
```

```

    });

    if (ChallengeName === "MFA_SETUP") {
        log("MFA setup is required.");
        return handleMfaSetup(Session, username);
    }

    if (ChallengeName === "SOFTWARE_TOKEN_MFA") {
        handleSoftwareTokenMfa(Session);
        log(`Run 'admin-respond-to-auth-challenge ${username} <totp>'`);
    }
} catch (err) {
    log(err);
}
};

export { adminInitiateAuthHandler };

const adminInitiateAuth = async ({
    clientId,
    userPoolId,
    username,
    password,
}) => {
    const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

    const command = new AdminInitiateAuthCommand({
        ClientId: clientId,
        UserPoolId: userPoolId,
        AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        AuthParameters: { USERNAME: username, PASSWORD: password },
    });

    return client.send(command);
};

import { log } from "../../../../../libs/utils/util-log.js";
import { adminRespondToAuthChallenge } from "../../../../../actions/admin-respond-to-auth-challenge.js";
import { getFirstEntry } from "../../../../../libs/utils/util-csv.js";
import { FILE_USER_POOLS } from "./constants.js";

const verifyUsername = (username) => {
    if (!username) {
        throw new Error(
            `Username is missing. It must be provided as an argument to the 'admin-respond-to-auth-challenge' command.`
        );
    }
};

const verifyTotp = (totp) => {
    if (!totp) {
        throw new Error(
            `Time-based one-time password (TOTP) is missing. It must be provided as an argument to the 'admin-respond-to-auth-challenge' command.`
        );
    }
};

const storeAccessToken = (token) => {
    process.env.AccessToken = token;
};

const adminRespondToAuthChallengeHandler = async (commands) => {
    const [_, username, totp] = commands;
}

```

```
try {
    verifyUsername(username);
    verifyTotp(totp);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    const session = process.env.SESSION;

    const { AuthenticationResult } = await adminRespondToAuthChallenge({
        clientId,
        userPoolId,
        username,
        totp,
        session,
    });

    storeAccessToken(AuthenticationResult.AccessToken);

    log("Successfully authenticated.");
} catch (err) {
    log(err);
}
};

export { adminRespondToAuthChallenge };

const respondToAuthChallenge = async ({
    clientId,
    username,
    session,
    userPoolId,
    code,
}) => {
    const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

    const command = new RespondToAuthChallengeCommand({
        ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
        ChallengeResponses: {
            SOFTWARE_TOKEN_MFA_CODE: code,
            USERNAME: username,
        },
        ClientId: clientId,
        UserPoolId: userPoolId,
        Session: session,
    });

    return client.send(command);
};

import { log } from "../../libs/utils/util-log.js";
import { verifySoftwareToken } from "../../actions/verify-software-token.js";

const validateTotp = (totp) => {
    if (!totp) {
        throw new Error(
            `Time-based one-time password (TOTP) must be provided to the 'validate-software-token' command.
        );
    }
};
const verifySoftwareTokenHandler = async (commands) => {
    const [_, totp] = commands;

    try {
        validateTotp(totp);
```

```
    log("Verifying TOTP.");
    await verifySoftwareToken(totp);
    log("TOTP Verified. Run 'admin-initiate-auth' again to sign-in.");
} catch (err) {
    console.log(err);
}
};

export { verifySoftwareTokenHandler };

const verifySoftwareToken = async (totp) => {
    const client = createClientForDefaultRegion(CognitoIdentityProviderClient);
    // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
    const session = process.env.SESSION;

    if (!session) {
        throw new Error(
            "Missing a valid Session. Did you run 'admin-initiate-auth'?"
        );
    }

    const command = new VerifySoftwareTokenCommand({
        Session: session,
        UserCode: totp,
    });

    return client.send(command);
};
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.
  - [Admin GetUser](#)
  - [AdminInitiateAuth](#)
  - [AdminRespondToAuthChallenge](#)
  - [AssociateSoftwareToken](#)
  - [ConfirmDevice](#)
  - [ConfirmSignUp](#)
  - [InitiateAuth](#)
  - [ListUsers](#)
  - [ResendConfirmationCode](#)
  - [RespondToAuthChallenge](#)
  - [SignUp](#)
  - [VerifySoftwareToken](#)

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
```

Before running this Kotlin code example, set up your development environment, including your credentials.

For more information, see the following documentation:  
<https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html>

TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS CDK) script provided in this GitHub repo at `resources/cdk/cognito_scenario_user_pool_with_mfa`.

This code example performs the following operations:

1. Invokes the `signUp` method to sign up a user.
  2. Invokes the `admin GetUser` method to get the user's confirmation status.
  3. Invokes the `ResendConfirmationCode` method if the user requested another code.
  4. Invokes the `confirmSignUp` method.
  5. Invokes the `initiateAuth` to sign in. This results in being prompted to set up TOTP (time-based one-time password). (The response is "ChallengeName": "MFA\_SETUP").
  6. Invokes the `AssociateSoftwareToken` method to generate a TOTP MFA private key. This can be used with Google Authenticator.
  7. Invokes the `VerifySoftwareToken` method to verify the TOTP and register for MFA.
  8. Invokes the `AdminInitiateAuth` to sign in again. This results in being prompted to submit a TOTP (Response: "ChallengeName": "SOFTWARE\_TOKEN\_MFA").
  9. Invokes the `AdminRespondToAuthChallenge` to get back a token.
- \*/

```
suspend fun main(args: Array<String>) {  
  
    val usage = """  
        Usage:  
            <clientId> <poolId>  
        Where:  
            clientId - The app client Id value that you can get from the AWS CDK  
script.  
            poolId - The pool Id that you can get from the AWS CDK script.  
        """  
  
    if (args.size != 2) {  
        println(usage)  
        exitProcess(1)  
    }  
  
    val clientId = args[0]  
    val poolId = args[1]  
  
    // Use the console to get data from the user.  
    println("**** Enter your use name")  
    val in0b = Scanner(System.`in`)  
    val userName = in0b.nextLine()  
    println(userName)  
  
    println("**** Enter your password")  
    val password: String = in0b.nextLine()  
  
    println("**** Enter your email")  
    val email = in0b.nextLine()  
  
    println("**** Signing up $userName")  
    signUp(clientId, userName, password, email)  
  
    println("**** Getting $userName in the user pool")  
    getAdminUser(userName, poolId)  
  
    println("**** Conformation code sent to $userName. Would you like to send a new  
code? (Yes/No)")
```

```
val ans = inOb.nextLine()

if (ans.compareTo("Yes") == 0) {
    println("**** Sending a new confirmation code")
    resendConfirmationCode(clientId, userName)
}
println("**** Enter the confirmation code that was emailed")
val code = inOb.nextLine()
confirmSignUp(clientId, code, userName)

println("**** Rechecking the status of $userName in the user pool")
getAdminUser(userName, poolId)

val authResponse = initiateAuth(clientId, userName, password)
val mySession = authResponse.session
val newSession = getSecretForAppMFA(mySession)
println("**** Enter the 6-digit code displayed in Google Authenticator")
val myCode = inOb.nextLine()

// Verify the TOTP and register for MFA.
verifyTOTP(newSession, myCode)
println("**** Re-enter a 6-digit code displayed in Google Authenticator")
val mfaCode: String = inOb.nextLine()
val authResponse1 = initiateAuth(clientId, userName, password)
val session2 = authResponse1.session
adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun resendConfirmationCode(clientIdVal: String?, userNameVal: String?) {

    val codeRequest = ResendConfirmationCodeRequest {
        clientId = clientIdVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
        (response.codeDeliveryDetails?.deliveryMedium))
    }
}

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(userName: String, clientIdVal: String?,
mfaCode: String, sessionVal: String?) {

    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponses0b = mutableMapOf<String, String>()
    challengeResponses0b["USERNAME"] = userName
    challengeResponses0b["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val respondToAuthChallengeRequest = RespondToAuthChallengeRequest {
        challengeName = ChallengeNameType.SoftwareTokenMfa
        clientId = clientIdVal
        challengeResponses = challengeResponses0b
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val respondToAuthChallengeResult =
        identityProviderClient.respondToAuthChallenge(respondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult() ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

```
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(sessionVal: String?, codeVal: String?) {

    val tokenRequest = VerifySoftwareTokenRequest {
        userCode = codeVal
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val verifyResponse =
            identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {

    val softwareTokenRequest = AssociateSoftwareTokenRequest {
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val tokenResponse =
            identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}

suspend fun initiateAuth(clientIdVal: String?, userNameVal: String, passwordVal: String): InitiateAuthResponse {

    val authParas = mutableMapOf <String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest = InitiateAuthRequest {
        clientId = clientIdVal
        authParameters = authParas
        authFlow = AuthFlowType.UserPasswordAuth
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val response = identityProviderClient.initiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}

suspend fun confirmSignUp(clientIdVal: String?, codeVal: String?, userNameVal: String?) {

    val signUpRequest = ConfirmSignUpRequest {
        clientId = clientIdVal
        confirmationCode = codeVal
        username = userNameVal
    }
}
```

```
CognitoIdentityProviderClient { region = "us-east-1" }.use
{ identityProviderClient ->
    identityProviderClient.confirmSignUp(signUpRequest)
    println("$userNameVal was confirmed")
}
}

suspend fun getAdminUser(userNameVal: String?, poolIdVal: String?) {
    val userRequest = Admin GetUserRequest {
        username = userNameVal
        userPoolId = poolIdVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        val response = identityProviderClient.admin GetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}

suspend fun signUp(clientIdVal: String?, userNameVal: String?, passwordVal: String?, emailVal: String?) {
    val userAttrs = AttributeType {
        name = "email"
        value = emailVal
    }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)

    val signUpRequest = SignUpRequest {
        userAttributes = userAttrsList
        username = userNameVal
        clientId = clientIdVal
        password = passwordVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
    { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - [Admin GetUser](#)
  - [Admin Initiate Auth](#)
  - [Admin Respond To Auth Challenge](#)
  - [Associate Software Token](#)
  - [Confirm Device](#)
  - [Confirm Sign Up](#)
  - [Initiate Auth](#)
  - [List Users](#)
  - [Resend Confirmation Code](#)
  - [Respond To Auth Challenge](#)
  - [Sign Up](#)
  - [Verify Software Token](#)

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

In addition to the previously listed steps, this example also registers an MFA device to be tracked by Amazon Cognito and shows you how to sign in by using a password and information from the tracked device. This avoids the need to enter a new MFA code.

```
def run_scenario(cognito_idp_client, user_pool_id, client_id):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the Amazon Cognito user signup with MFA demo.")
    print('*'*88)

    cog_wrapper = CognitoIdentityProviderWrapper(cognito_idp_client, user_pool_id,
                                                client_id)

    user_name = q.ask("Let's sign up a new user. Enter a user name: ", q.non_empty)
    password = q.ask("Enter a password for the user: ", q.non_empty)
    email = q.ask("Enter a valid email address that you own: ", q.non_empty)
    confirmed = cog_wrapper.sign_up_user(user_name, password, email)
    while not confirmed:
        print(f"User {user_name} requires confirmation. Check {email} for "
              f"a verification code.")
        confirmation_code = q.ask("Enter the confirmation code from the email: ")
        if not confirmation_code:
            if q.ask("Do you need another confirmation code (y/n)? ", q.is_yesno):
                delivery = cog_wrapper.resend_confirmation(user_name)
                print(f"Confirmation code sent by {delivery['DeliveryMedium']} "
                      f"to {delivery['Destination']}.")

        else:
            confirmed = cog_wrapper.confirm_user_sign_up(user_name,
                                                        confirmation_code)
    print(f"User {user_name} is confirmed and ready to use.")
    print('*'*88)

    print("Let's get a list of users in the user pool.")
    q.ask("Press Enter when you're ready.")
    users = cog_wrapper.list_users()
    if users:
        print(f"Found {len(users)} users:")
        pp(users)
    else:
        print("No users found.")
    print('*'*88)

    print("Let's sign in and get an access token.")
    auth_tokens = None
    challenge = 'ADMIN_USER_PASSWORD_AUTH'
    response = {}
    while challenge is not None:
        if challenge == 'ADMIN_USER_PASSWORD_AUTH':
            response = cog_wrapper.start_sign_in(user_name, password)
            challenge = response['ChallengeName']
        elif response['ChallengeName'] == 'MFA_SETUP':
            print("First, we need to set up an MFA application.")
            qr_img = qrcode.make(
                f"otpauth://totp/{user_name}?secret={response['SecretCode']}"))
            qr_img.save("qr.png")
```

```
        q.ask("Press Enter to see a QR code on your screen. Scan it into an MFA
"
      "application, such as Google Authenticator.")
      webbrowser.open("qr.png")
      mfa_code = q.ask(
        "Enter the verification code from your MFA application: ",
      q.non_empty)
      response = cog_wrapper.verify_mfa(response['Session'], mfa_code)
      print(f"MFA device setup {response['Status']}"))
      print("Now that an MFA application is set up, let's sign in again.")
      print("You might have to wait a few seconds for a new MFA code to
appear in "
        "your MFA application.")
      challenge = 'ADMIN_USER_PASSWORD_AUTH'
      elif response['ChallengeName'] == 'SOFTWARE_TOKEN_MFA':
        auth_tokens = None
        while auth_tokens is None:
          mfa_code = q.ask(
            "Enter a verification code from your MFA application: ",
      q.non_empty)
          auth_tokens = cog_wrapper.respond_to_mfa_challenge(
            user_name, response['Session'], mfa_code)
          print(f"You're signed in as {user_name}.")
          print("Here's your access token:")
          pp(auth_tokens['AccessToken'])
          print("And your device information:")
          pp(auth_tokens['NewDeviceMetadata'])
          challenge = None
        else:
          raise Exception(f"Got unexpected challenge
{response['ChallengeName']}"))
      print('-*88)

      device_group_key = auth_tokens['NewDeviceMetadata']['DeviceGroupKey']
      device_key = auth_tokens['NewDeviceMetadata']['DeviceKey']
      device_password = base64.standard_b64encode(os.urandom(40)).decode('utf-8')

      print("Let's confirm your MFA device so you don't have re-enter MFA tokens for
it.")
      q.ask("Press Enter when you're ready.")
      cog_wrapper.confirm_mfa_device(
        user_name, device_key, device_group_key, device_password,
        auth_tokens['AccessToken'], aws_srp)
      print(f"Your device {device_key} is confirmed.")
      print('-*88)

      print(f"Now let's sign in as {user_name} from your confirmed device
{device_key}.\n"
        f"Because this device is tracked by Amazon Cognito, you won't have to re-
enter an MFA code.")
      q.ask("Press Enter when ready.")
      auth_tokens = cog_wrapper.sign_in_with_tracked_device(
        user_name, password, device_key, device_group_key, device_password,
      aws_srp)
      print("You're signed in. Your access token is:")
      pp(auth_tokens['AccessToken'])
      print('-*88)

      print("Don't forget to delete your user pool when you're done with this
example.")
      print("\nThanks for watching!")
      print('-*88)

def main():
  parser = argparse.ArgumentParser()
```

```
        description="Shows how to sign up a new user with Amazon Cognito and
associate "
                "the user with an MFA application for multi-factor
authentication.")
parser.add_argument('user_pool_id', help="The ID of the user pool to use for
the example.")
parser.add_argument('client_id', help="The ID of the client application to use
for the example.")
args = parser.parse_args()
try:
    run_scenario(boto3.client('cognito-identity'), args.user_pool_id,
args.client_id)
except Exception:
    logging.exception("Something went wrong with the demo.")

if __name__ == '__main__':
    main()
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [AdminGetUser](#)
  - [AdminInitiateAuth](#)
  - [AdminRespondToAuthChallenge](#)
  - [AssociateSoftwareToken](#)
  - [ConfirmDevice](#)
  - [ConfirmSignUp](#)
  - [InitiateAuth](#)
  - [ListUsers](#)
  - [ResendConfirmationCode](#)
  - [RespondToAuthChallenge](#)
  - [SignUp](#)
  - [VerifySoftwareToken](#)

## Code examples for Amazon Cognito Sync using AWS SDKs

The following code examples show how to use Amazon Cognito Sync with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon Cognito Sync using AWS SDKs \(p. 274\)](#)
  - [Get a list of identity pools registered with Amazon Cognito using an AWS SDK \(p. 275\)](#)

## Actions for Amazon Cognito Sync using AWS SDKs

The following code examples show how to use Amazon Cognito Sync with AWS SDKs. Each example calls an individual service function.

### Examples

- [Get a list of identity pools registered with Amazon Cognito using an AWS SDK \(p. 275\)](#)

## Get a list of identity pools registered with Amazon Cognito using an AWS SDK

The following code example shows how to list Amazon Cognito identity pools.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client
        .list_identity_pool_usage()
        .max_results(10)
        .send()
        .await?;

    if let Some(pools) = response.identity_pool_usages() {
        println!("Identity pools:");

        for pool in pools {
            println!(
                "  Identity pool ID: {}",
                pool.identity_pool_id().unwrap_or_default()
            );
            println!(
                "  Data storage: {}",
                pool.data_storage().unwrap_or_default()
            );
            println!(
                "  Sync sessions count: {}",
                pool.sync_sessions_count().unwrap_or_default()
            );
            println!(
                "  Last modified: {}",
                pool.last_modified_date().unwrap().to_chrono_utc()
            );
            println!();
        }
    }

    println!("Next token: {}", response.next_token().unwrap_or_default());
    Ok(())
}
```

- For API details, see [ListIdentityPoolUsage in AWS SDK for Rust API reference](#).

# Code examples for Amazon Comprehend using AWS SDKs

The following code examples show how to use Amazon Comprehend with an AWS software development kit (SDK).

## Code examples

- [Actions for Amazon Comprehend using AWS SDKs \(p. 276\)](#)
  - [Create an Amazon Comprehend document classifier using an AWS SDK \(p. 277\)](#)
  - [Delete an Amazon Comprehend document classifier using an AWS SDK \(p. 279\)](#)
  - [Describe an Amazon Comprehend document classification job using an AWS SDK \(p. 279\)](#)
  - [Describe an Amazon Comprehend document classifier using an AWS SDK \(p. 280\)](#)
  - [Describe an Amazon Comprehend topic modeling job using an AWS SDK \(p. 281\)](#)
  - [Detect entities in a document with Amazon Comprehend using an AWS SDK \(p. 282\)](#)
  - [Detect key phrases in a document with Amazon Comprehend using an AWS SDK \(p. 284\)](#)
  - [Detect personally identifiable information in a document with Amazon Comprehend using an AWS SDK \(p. 286\)](#)
  - [Detect syntactical elements of a document with Amazon Comprehend using an AWS SDK \(p. 288\)](#)
  - [Detect the dominant language in a document with Amazon Comprehend using an AWS SDK \(p. 290\)](#)
  - [Detect the sentiment of a document with Amazon Comprehend using an AWS SDK \(p. 292\)](#)
  - [List Amazon Comprehend document classification jobs using an AWS SDK \(p. 294\)](#)
  - [List Amazon Comprehend document classifiers using an AWS SDK \(p. 295\)](#)
  - [List Amazon Comprehend topic modeling jobs using an AWS SDK \(p. 295\)](#)
  - [Start an Amazon Comprehend document classification job using an AWS SDK \(p. 296\)](#)
  - [Start an Amazon Comprehend topic modeling job using an AWS SDK \(p. 297\)](#)
- [Scenarios for Amazon Comprehend using AWS SDKs \(p. 300\)](#)
  - [Detect document elements with Amazon Comprehend and an AWS SDK \(p. 300\)](#)
  - [Run an Amazon Comprehend topic modeling job on sample data using an AWS SDK \(p. 304\)](#)
  - [Train a custom Amazon Comprehend classifier and classify documents using an AWS SDK \(p. 306\)](#)
- [Cross-service examples for Amazon Comprehend using AWS SDKs \(p. 313\)](#)
  - [Build an Amazon Transcribe streaming app \(p. 313\)](#)
  - [Create an Amazon Lex Chatbot within a web application to engage your web site visitors \(p. 314\)](#)
  - [Create a web application that sends and retrieves messages by using the Amazon Simple Queue Service API \(p. 315\)](#)
  - [Detect entities in text extracted from an image using an AWS SDK \(p. 315\)](#)

## Actions for Amazon Comprehend using AWS SDKs

---

The following code examples show how to use Amazon Comprehend with AWS SDKs. Each example calls an individual service function.

## Examples

- [Create an Amazon Comprehend document classifier using an AWS SDK \(p. 277\)](#)
- [Delete an Amazon Comprehend document classifier using an AWS SDK \(p. 279\)](#)
- [Describe an Amazon Comprehend document classification job using an AWS SDK \(p. 279\)](#)
- [Describe an Amazon Comprehend document classifier using an AWS SDK \(p. 280\)](#)
- [Describe an Amazon Comprehend topic modeling job using an AWS SDK \(p. 281\)](#)
- [Detect entities in a document with Amazon Comprehend using an AWS SDK \(p. 282\)](#)
- [Detect key phrases in a document with Amazon Comprehend using an AWS SDK \(p. 284\)](#)
- [Detect personally identifiable information in a document with Amazon Comprehend using an AWS SDK \(p. 286\)](#)
- [Detect syntactical elements of a document with Amazon Comprehend using an AWS SDK \(p. 288\)](#)
- [Detect the dominant language in a document with Amazon Comprehend using an AWS SDK \(p. 290\)](#)
- [Detect the sentiment of a document with Amazon Comprehend using an AWS SDK \(p. 292\)](#)
- [List Amazon Comprehend document classification jobs using an AWS SDK \(p. 294\)](#)
- [List Amazon Comprehend document classifiers using an AWS SDK \(p. 295\)](#)
- [List Amazon Comprehend topic modeling jobs using an AWS SDK \(p. 295\)](#)
- [Start an Amazon Comprehend document classification job using an AWS SDK \(p. 296\)](#)
- [Start an Amazon Comprehend topic modeling job using an AWS SDK \(p. 297\)](#)

## Create an Amazon Comprehend document classifier using an AWS SDK

The following code examples show how to create an Amazon Comprehend document classifier.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createDocumentClassifier(ComprehendClient comClient, String
dataAccessRoleArn, String s3Uri, String documentClassifierName){

    try {
        DocumentClassifierInputDataConfig config =
DocumentClassifierInputDataConfig.builder()
            .s3Uri(s3Uri)
            .build();

        CreateDocumentClassifierRequest createDocumentClassifierRequest =
CreateDocumentClassifierRequest.builder()
            .documentClassifierName(documentClassifierName)
            .dataAccessRoleArn(dataAccessRoleArn)
            .languageCode("en")
            .inputDataConfig(config)
            .build();

        CreateDocumentClassifierResponse createDocumentClassifierResult =
comClient.createDocumentClassifier(createDocumentClassifierRequest);
        String documentClassifierArn =
createDocumentClassifierResult.documentClassifierArn();
```

```
        System.out.println("Document Classifier ARN: " +  
documentClassifierArn);  
  
    } catch (ComprehendException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [CreateDocumentClassifier](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendClassifier:  
    """Encapsulates an Amazon Comprehend custom classifier."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
        self.classifier_arn = None  
  
    def create(  
        self, name, language_code, training_bucket, training_key,  
        data_access_role_arn, mode):  
        """  
        Creates a custom classifier. After the classifier is created, it  
        immediately  
        starts training on the data found in the specified Amazon S3 bucket.  
        Training  
        can take 30 minutes or longer. The `describe_document_classifier` function  
        can be used to get training status and returns a status of TRAINED when the  
        classifier is ready to use.  
  
        :param name: The name of the classifier.  
        :param language_code: The language the classifier can operate on.  
        :param training_bucket: The Amazon S3 bucket that contains the training  
        data.  
        :param training_key: The prefix used to find training data in the training  
        bucket. If multiple objects have the same prefix, all  
        of them are used.  
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that  
        grants Comprehend permission to read from the  
        training bucket.  
        :return: The ARN of the newly created classifier.  
        """  
        try:  
            response = self.comprehend_client.create_document_classifier(  
                DocumentClassifierName=name,  
                LanguageCode=language_code,  
                InputDataConfig={'S3Uri': f's3://{training_bucket}/  
{training_key}'},  
                DataAccessRoleArn=data_access_role_arn,  
                Mode=mode.value)  
            self.classifier_arn = response['DocumentClassifierArn']
```

```
        logger.info("Started classifier creation. Arn is: %s.",  
self.classifier_arn)  
    except ClientError:  
        logger.exception("Couldn't create classifier %s.", name)  
        raise  
    else:  
        return self.classifier_arn
```

- For API details, see [CreateDocumentClassifier](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an Amazon Comprehend document classifier using an AWS SDK

The following code example shows how to delete an Amazon Comprehend document classifier.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendClassifier:  
    """Encapsulates an Amazon Comprehend custom classifier."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
        self.classifier_arn = None  
  
    def delete(self):  
        """  
        Deletes the classifier.  
        """  
        try:  
            self.comprehend_client.delete_document_classifier(  
                DocumentClassifierArn=self.classifier_arn)  
            logger.info("Deleted classifier %s.", self.classifier_arn)  
            self.classifier_arn = None  
        except ClientError:  
            logger.exception("Couldn't deleted classifier %s.",  
self.classifier_arn)  
            raise
```

- For API details, see [DeleteDocumentClassifier](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe an Amazon Comprehend document classification job using an AWS SDK

The following code example shows how to describe an Amazon Comprehend document classification job.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendClassifier:  
    """Encapsulates an Amazon Comprehend custom classifier."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
        self.classifier_arn = None  
  
    def describe_job(self, job_id):  
        """  
        Gets metadata about a classification job.  
  
        :param job_id: The ID of the job to look up.  
        :return: Metadata about the job.  
        """  
        try:  
            response = self.comprehend_client.describe_document_classification_job(  
                JobId=job_id)  
            job = response['DocumentClassificationJobProperties']  
            logger.info("Got classification job %s.", job['JobName'])  
        except ClientError:  
            logger.exception("Couldn't get classification job %s.", job_id)  
            raise  
        else:  
            return job
```

- For API details, see [DescribeDocumentClassificationJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe an Amazon Comprehend document classifier using an AWS SDK

The following code example shows how to describe an Amazon Comprehend document classifier.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendClassifier:  
    """Encapsulates an Amazon Comprehend custom classifier."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
        self.classifier_arn = None
```

```
def describe(self, classifier_arn=None):
    """
    Gets metadata about a custom classifier, including its current status.

    :param classifier_arn: The ARN of the classifier to look up.
    :return: Metadata about the classifier.
    """
    if classifier_arn is not None:
        self.classifier_arn = classifier_arn
    try:
        response = self.comprehend_client.describe_document_classifier(
            DocumentClassifierArn=self.classifier_arn)
        classifier = response['DocumentClassifierProperties']
        logger.info("Got classifier %s.", self.classifier_arn)
    except ClientError:
        logger.exception("Couldn't get classifier %s.", self.classifier_arn)
        raise
    else:
        return classifier
```

- For API details, see [DescribeDocumentClassifier](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe an Amazon Comprehend topic modeling job using an AWS SDK

The following code example shows how to describe an Amazon Comprehend topic modeling job.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendTopicModeler:
    """
    Encapsulates a Comprehend topic modeler.
    """
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client

    def describe_job(self, job_id):
        """
        Gets metadata about a topic modeling job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_topics_detection_job(
                JobId=job_id)
            job = response['TopicsDetectionJobProperties']
            logger.info("Got topic detection job %s.", job_id)
        except ClientError:
            logger.exception("Couldn't get topic detection job %s.", job_id)
            raise
        else:
```

```
    return job
```

- For API details, see [DescribeTopicsDetectionJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect entities in a document with Amazon Comprehend using an AWS SDK

The following code examples show how to detect entities in a document with Amazon Comprehend.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

///<summary>
/// This example shows how to use the AmazonComprehend service detect any
/// entities in submitted text. This example was created using the AWS SDK
/// for .NET 3.7 and .NET Core 5.0.
///</summary>
public static class DetectEntities
{
    ///<summary>
    /// The main method calls the DetectEntitiesAsync method to find any
    /// entities in the sample code.
    ///</summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle";

        var comprehendClient = new AmazonComprehendClient();

        Console.WriteLine("Calling DetectEntities\n");
        var detectEntitiesRequest = new DetectEntitiesRequest()
        {
            Text = text,
            LanguageCode = "en",
        };
        var detectEntitiesResponse = await
comprehendClient.DetectEntitiesAsync(detectEntitiesRequest);

        foreach (var e in detectEntitiesResponse.Entities)
        {
            Console.WriteLine($"Text: {e.Text}, Type: {e.Type}, Score:
{e.Score}, BeginOffset: {e.BeginOffset}, EndOffset: {e.EndOffset}");
        }

        Console.WriteLine("Done");
    }
}
```

- For API details, see [DetectEntities](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectAllEntities(ComprehendClient comClient, String text) {  
  
    try {  
        DetectEntitiesRequest detectEntitiesRequest =  
        DetectEntitiesRequest.builder()  
            .text(text)  
            .languageCode("en")  
            .build();  
  
        DetectEntitiesResponse detectEntitiesResult =  
        comClient.detectEntities(detectEntitiesRequest);  
        List<Entity> entList = detectEntitiesResult.entities();  
        for (Entity entity : entList) {  
            System.out.println("Entity text is " + entity.text());  
        }  
  
    } catch (ComprehendException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DetectEntities](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendDetect:  
    """Encapsulates Comprehend detection functions."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
  
    def detect_entities(self, text, language_code):  
        """  
        Detects entities in a document. Entities can be things like people and  
        places  
        or other common terms.  
  
        :param text: The document to inspect.  
        :param language_code: The language of the document.  
        """
```

```
:return: The list of entities along with their confidence scores.  
"""  
try:  
    response = self.comprehend_client.detect_entities(  
        Text=text, LanguageCode=language_code)  
    entities = response['Entities']  
    logger.info("Detected %s entities.", len(entities))  
except ClientError:  
    logger.exception("Couldn't detect entities.")  
    raise  
else:  
    return entities
```

- For API details, see [DetectEntities](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect key phrases in a document with Amazon Comprehend using an AWS SDK

The following code examples show how to detect key phrases in a document with Amazon Comprehend.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Threading.Tasks;  
using Amazon.Comprehend;  
using Amazon.Comprehend.Model;  
  
/// <summary>  
/// This example shows how to use the Amazon Comprehend service to  
/// search text for key phrases. It was created using the AWS SDK for  
/// .NET version 3.7 and .NET Core 5.0.  
/// </summary>  
public static class DetectKeyPhrase  
{  
    /// <summary>  
    /// This method calls the Amazon Comprehend method DetectKeyPhrasesAsync  
    /// to detect any key phrases in the sample text.  
    /// </summary>  
    public static async Task Main()  
    {  
        string text = "It is raining today in Seattle";  
  
        var comprehendClient = new  
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);  
  
        // Call DetectKeyPhrases API  
        Console.WriteLine("Calling DetectKeyPhrases");  
        var detectKeyPhrasesRequest = new DetectKeyPhrasesRequest()  
        {  
            Text = text,  
            LanguageCode = "en",  
        };  
        var detectKeyPhrasesResponse = await  
comprehendClient.DetectKeyPhrasesAsync(detectKeyPhrasesRequest);
```

```
foreach (var kp in detectKeyPhrasesResponse.KeyPhrases)
{
    Console.WriteLine($"Text: {kp.Text}, Score: {kp.Score},
BeginOffset: {kp.BeginOffset}, EndOffset: {kp.EndOffset}");
}

Console.WriteLine("Done");
}
```

- For API details, see [DetectKeyPhrases](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectAllKeyPhrases(ComprehendClient comClient, String text)
{
    try {
        DetectKeyPhrasesRequest detectKeyPhrasesRequest =
        DetectKeyPhrasesRequest.builder()
            .text(text)
            .languageCode("en")
            .build();

        DetectKeyPhrasesResponse detectKeyPhrasesResult =
        comClient.detectKeyPhrases(detectKeyPhrasesRequest);
        List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();
        for (KeyPhrase keyPhrase : phraseList) {
            System.out.println("Key phrase text is " + keyPhrase.text());
        }
    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DetectKeyPhrases](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""
    def __init__(self, comprehend_client):
```

```
"""
:param comprehend_client: A Boto3 Comprehend client.
"""
self.comprehend_client = comprehend_client

def detect_key_phrases(self, text, language_code):
    """
    Detects key phrases in a document. A key phrase is typically a noun and its
    modifiers.

    :param text: The document to inspect.
    :param language_code: The language of the document.
    :return: The list of key phrases along with their confidence scores.
    """
    try:
        response = self.comprehend_client.detect_key_phrases(
            Text=text, LanguageCode=language_code)
        phrases = response['KeyPhrases']
        logger.info("Detected %s phrases.", len(phrases))
    except ClientError:
        logger.exception("Couldn't detect phrases.")
        raise
    else:
        return phrases
```

- For API details, see [DetectKeyPhrases](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect personally identifiable information in a document with Amazon Comprehend using an AWS SDK

The following code examples show how to detect personally identifiable information (PII) in a document with Amazon Comprehend.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to use the Amazon Comprehend service to find
/// personally identifiable information (PII) within text submitted to the
/// DetectPiiEntitiesAsync method. The example was created using the AWS
/// SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DetectingPII
{
    /// <summary>
    /// This method calls the DetectPiiEntitiesAsync method to locate any
    /// personally identifiable information within the supplied text.
    /// </summary>
    public static async Task Main()
```

```
{
    var comprehendClient = new AmazonComprehendClient();
    var text = @"Hello Paul Santos. The latest statement for your
                credit card account 1111-0000-1111-0000 was
                mailed to 123 Any Street, Seattle, WA 98109.";

    var request = new DetectPiiEntitiesRequest
    {
        Text = text,
        LanguageCode = "EN",
    };

    var response = await comprehendClient.DetectPiiEntitiesAsync(request);

    if (response.Entities.Count > 0)
    {
        foreach (var entity in response.Entities)
        {
            var entityValue = text.Substring(entity.BeginOffset,
entity.EndOffset - entity.BeginOffset);
            Console.WriteLine($"{entity.Type}: {entityValue}");
        }
    }
}
```

- For API details, see [DetectPiiEntities](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client

    def detect_pii(self, text, language_code):
        """
        Detects personally identifiable information (PII) in a document. PII can be
        things like names, account numbers, or addresses.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of PII entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_pii_entities(
                Text=text, LanguageCode=language_code)
            entities = response['Entities']
            logger.info("Detected %s PII entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect PII entities.")
```

```
    raise
else:
    return entities
```

- For API details, see [DetectPiiEntities](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect syntactical elements of a document with Amazon Comprehend using an AWS SDK

The following code examples show how to detect syntactical elements of a document with Amazon Comprehend.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to use Amazon Comprehend to detect syntax
/// elements by calling the DetectSyntaxAsync method. This example was
/// created using the AWS SDK for .NET 3.7 and .NET Core 5.0.
/// </summary>
public class DetectingSyntax
{
    /// <summary>
    /// This method calls DetectSyntaxAsync to identify the syntax elements
    /// in the sample text.
    /// </summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle";

        var comprehendClient = new AmazonComprehendClient();

        // Call DetectSyntax API
        Console.WriteLine("Calling DetectSyntaxAsync\n");
        var detectSyntaxRequest = new DetectSyntaxRequest()
        {
            Text = text,
            LanguageCode = "en",
        };
        DetectSyntaxResponse detectSyntaxResponse = await
comprehendClient.DetectSyntaxAsync(detectSyntaxRequest);
        foreach (SyntaxToken s in detectSyntaxResponse.SyntaxTokens)
        {
            Console.WriteLine($"Text: {s.Text}, PartOfSpeech:
{s.PartOfSpeech.Tag}, BeginOffset: {s.BeginOffset}, EndOffset: {s.EndOffset}");
        }

        Console.WriteLine("Done");
    }
}
```

- For API details, see [DetectSyntax](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectAllSyntax(ComprehendClient comClient, String text){  
  
    try {  
        DetectSyntaxRequest detectSyntaxRequest = DetectSyntaxRequest.builder()  
            .text(text)  
            .languageCode("en")  
            .build();  
  
        DetectSyntaxResponse detectSyntaxResult =  
comClient.detectSyntax(detectSyntaxRequest);  
        List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();  
        for (SyntaxToken token : syntaxTokens) {  
            System.out.println("Language is " + token.text());  
            System.out.println("Part of speech is " +  
token.partOfSpeech().tagAsString());  
        }  
  
    } catch (ComprehendException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DetectSyntax](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendDetect:  
    """Encapsulates Comprehend detection functions."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
  
    def detect_syntax(self, text, language_code):  
        """  
        Detects syntactical elements of a document. Syntax tokens are portions of  
        text along with their use as parts of speech, such as nouns, verbs, and  
        
```

```
interjections.

:param text: The document to inspect.
:param language_code: The language of the document.
:return: The list of syntax tokens along with their confidence scores.
"""
try:
    response = self.comprehend_client.detect_syntax(
        Text=text, LanguageCode=language_code)
    tokens = response['SyntaxTokens']
    logger.info("Detected %s syntax tokens.", len(tokens))
except ClientError:
    logger.exception("Couldn't detect syntax.")
    raise
else:
    return tokens
```

- For API details, see [DetectSyntax](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect the dominant language in a document with Amazon Comprehend using an AWS SDK

The following code examples show how to detect the dominant language in a document with Amazon Comprehend.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example calls the Amazon Comprehend service to determine the
/// dominant language. The example was created using the AWS SDK for .NET
/// 3.7 and .NET Core 5.0.
/// </summary>
public static class DetectDominantLanguage
{
    /// <summary>
    /// Calls Amazon Comprehend to determine the dominant language used in
    /// the sample text.
    /// </summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle.";

        var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

        Console.WriteLine("Calling DetectDominantLanguage\n");
        var detectDominantLanguageRequest = new DetectDominantLanguageRequest()
        {
            Text = text,
```

```
};

    var detectDominantLanguageResponse = await
comprehendClient.DetectDominantLanguageAsync(detectDominantLanguageRequest);
    foreach (var dl in detectDominantLanguageResponse.Languages)
    {
        Console.WriteLine($"Language Code: {dl.LanguageCode}, Score:
{dl.Score}");
    }

    Console.WriteLine("Done");
}
}
```

- For API details, see [DetectDominantLanguage](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectTheDominantLanguage(ComprehendClient comClient, String
text){

    try {
        DetectDominantLanguageRequest request =
DetectDominantLanguageRequest.builder()
            .text(text)
            .build();

        DetectDominantLanguageResponse resp =
comClient.detectDominantLanguage(request);
        List<DominantLanguage> allLanList = resp.languages();
        for (DominantLanguage lang : allLanList) {
            System.out.println("Language is " + lang.languageCode());
        }
    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DetectDominantLanguage](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendDetect:
```

```
"""Encapsulates Comprehend detection functions."""
def __init__(self, comprehend_client):
    """
    :param comprehend_client: A Boto3 Comprehend client.
    """
    self.comprehend_client = comprehend_client

    def detect_languages(self, text):
        """
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_dominant_language(Text=text)
            languages = response['Languages']
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
            return languages
```

- For API details, see [DetectDominantLanguage](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect the sentiment of a document with Amazon Comprehend using an AWS SDK

The following code examples show how to detect the sentiment of a document with Amazon Comprehend.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to detect the overall sentiment of the supplied
/// text using the Amazon Comprehend service. The example was writing using
/// the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public static class DetectSentiment
{
    /// <summary>
    /// This method calls the DetetectSentimentAsync method to analyze the
    /// supplied text and determine the overal sentiment.
    /// </summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle";
```

```
var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

// Call DetectKeyPhrases API
Console.WriteLine("Calling DetectSentiment");
var detectSentimentRequest = new DetectSentimentRequest()
{
    Text = text,
    LanguageCode = "en",
};
var detectSentimentResponse = await
comprehendClient.DetectSentimentAsync(detectSentimentRequest);
Console.WriteLine($"Sentiment: {detectSentimentResponse.Sentiment}");
Console.WriteLine("Done");
}
```

- For API details, see [DetectSentiment](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectSentiments(ComprehendClient comClient, String text){

    try {
        DetectSentimentRequest detectSentimentRequest =
        DetectSentimentRequest.builder()
            .text(text)
            .languageCode("en")
            .build();

        DetectSentimentResponse detectSentimentResult =
        comClient.detectSentiment(detectSentimentRequest);
        System.out.println("The Neutral value is "
        +detectSentimentResult.sentimentScore().neutral());

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DetectSentiment](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendDetect:  
    """Encapsulates Comprehend detection functions."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
  
    def detect_sentiment(self, text, language_code):  
        """  
        Detects the overall sentiment expressed in a document. Sentiment can  
        be positive, negative, neutral, or a mixture.  
  
        :param text: The document to inspect.  
        :param language_code: The language of the document.  
        :return: The sentiments along with their confidence scores.  
        """  
        try:  
            response = self.comprehend_client.detect_sentiment(  
                Text=text, LanguageCode=language_code)  
            logger.info("Detected primary sentiment %s.", response['Sentiment'])  
        except ClientError:  
            logger.exception("Couldn't detect sentiment.")  
            raise  
        else:  
            return response
```

- For API details, see [DetectSentiment](#) in *AWS SDK for Python (Boto3) API Reference*.

## List Amazon Comprehend document classification jobs using an AWS SDK

The following code example shows how to list Amazon Comprehend document classification jobs.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendClassifier:  
    """Encapsulates an Amazon Comprehend custom classifier."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
        self.classifier_arn = None  
  
    def list_jobs(self):  
        """  
        Lists the classification jobs for the current account.  
  
        :return: The list of jobs.  
        """  
        try:
```

```
response = self.comprehend_client.list_document_classification_jobs()
jobs = response['DocumentClassificationJobPropertiesList']
logger.info("Got %s document classification jobs.", len(jobs))
except ClientError:
    logger.exception("Couldn't get document classification jobs.", )
    raise
else:
    return jobs
```

- For API details, see [ListDocumentClassificationJobs in AWS SDK for Python \(Boto3\) API Reference](#).

## List Amazon Comprehend document classifiers using an AWS SDK

The following code example shows how to list Amazon Comprehend document classifiers.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None

    def list(self):
        """
        Lists custom classifiers for the current account.

        :return: The list of classifiers.
        """
        try:
            response = self.comprehend_client.list_document_classifiers()
            classifiers = response['DocumentClassifierPropertiesList']
            logger.info("Got %s classifiers.", len(classifiers))
        except ClientError:
            logger.exception("Couldn't get classifiers.", )
            raise
        else:
            return classifiers
```

- For API details, see [ListDocumentClassifiers in AWS SDK for Python \(Boto3\) API Reference](#).

## List Amazon Comprehend topic modeling jobs using an AWS SDK

The following code example shows how to list Amazon Comprehend topic modeling jobs.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendTopicModeler:  
    """Encapsulates a Comprehend topic modeler."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
  
    def list_jobs(self):  
        """  
        Lists topic modeling jobs for the current account.  
        :return: The list of jobs.  
        """  
        try:  
            response = self.comprehend_client.list_topics_detection_jobs()  
            jobs = response['TopicsDetectionJobPropertiesList']  
            logger.info("Got %s topic detection jobs.", len(jobs))  
        except ClientError:  
            logger.exception("Couldn't get topic detection jobs.")  
            raise  
        else:  
            return jobs
```

- For API details, see [ListTopicsDetectionJobs](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start an Amazon Comprehend document classification job using an AWS SDK

The following code example shows how to start an Amazon Comprehend document classification job.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendClassifier:  
    """Encapsulates an Amazon Comprehend custom classifier."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
        self.classifier_arn = None  
  
    def start_job(  
        self, job_name, input_bucket, input_key, input_format, output_bucket,
```

```
        output_key, data_access_role_arn):
"""
Starts a classification job. The classifier must be trained or the job
will fail. Input is read from the specified Amazon S3 input bucket and
written to the specified output bucket. Output data is stored in a tar
archive compressed in gzip format. The job runs asynchronously, so you can
call `describe_document_classification_job` to get job status until it
returns a status of SUCCEEDED.

:param job_name: The name of the job.
:param input_bucket: The Amazon S3 bucket that contains input data.
:param input_key: The prefix used to find input data in the input
    bucket. If multiple objects have the same prefix, all
    of them are used.
:param input_format: The format of the input data, either one document per
    file or one document per line.
:param output_bucket: The Amazon S3 bucket where output data is written.
:param output_key: The prefix prepended to the output data.
:param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
    grants Comprehend permission to read from the
    input bucket and write to the output bucket.
:return: Information about the job, including the job ID.
"""
try:
    response = self.comprehend_client.start_document_classification_job(
        DocumentClassifierArn=self.classifier_arn,
        JobName=job_name,
        InputDataConfig={
            'S3Uri': f's3://{input_bucket}/{input_key}',
            'InputFormat': input_format.value,
            'OutputDataConfig': {'S3Uri': f's3://{output_bucket}/{output_key}'},
            'DataAccessRoleArn': data_access_role_arn
        }
    )
    logger.info(
        "Document classification job %s is %s.", job_name,
        response['JobStatus']
    )
except ClientError:
    logger.exception("Couldn't start classification job %s.", job_name)
    raise
else:
    return response
```

- For API details, see [StartDocumentClassificationJob in AWS SDK for Python \(Boto3\) API Reference](#).

## Start an Amazon Comprehend topic modeling job using an AWS SDK

The following code examples show how to start an Amazon Comprehend topic modeling job.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
```

```
using Amazon.Comprehend.Model;

///<summary>
/// This example scans the documents in an Amazon Simple Storage Service
/// (Amazon S3) bucket and analyzes it for topics. The results are stored
/// in another bucket and then the resulting job properties are displayed
/// on the screen. This example was created using the AWS SDK for .NET
/// version 3.7 and .NET Core version 5.0.
///</summary>
public static class TopicModeling
{
    ///<summary>
    /// This methos calls a topic detection job by calling the Amazon
    /// Comprehend StartTopicsDetectionJobRequest.
    ///</summary>
    public static async Task Main()
    {
        var comprehendClient = new AmazonComprehendClient();

        string inputS3Uri = "s3://input bucket/input path";
        InputFormat inputDocFormat = InputFormat.ONE_DOC_PER_FILE;
        string outputS3Uri = "s3://output bucket/output path";
        string dataAccessRoleArn = "arn:aws:iam::account ID:role/data access
role";
        int numberofTopics = 10;

        var startTopicsDetectionJobRequest = new
StartTopicsDetectionJobRequest()
        {
            InputDataConfig = new InputDataConfig()
            {
                S3Uri = inputS3Uri,
                InputFormat = inputDocFormat,
            },
            OutputDataConfig = new OutputDataConfig()
            {
                S3Uri = outputS3Uri,
            },
            DataAccessRoleArn = dataAccessRoleArn,
            NumberOfTopics = numberofTopics,
        };

        var startTopicsDetectionJobResponse = await
comprehendClient.StartTopicsDetectionJobAsync(startTopicsDetectionJobRequest);

        var jobId = startTopicsDetectionJobResponse.JobId;
        Console.WriteLine("JobId: " + jobId);

        var describeTopicsDetectionJobRequest = new
DescribeTopicsDetectionJobRequest()
        {
            JobId = jobId,
        };

        var describeTopicsDetectionJobResponse = await
comprehendClient.DescribeTopicsDetectionJobAsync(describeTopicsDetectionJobRequest);

PrintJobProperties(describeTopicsDetectionJobResponse.TopicsDetectionJobProperties);

        var listTopicsDetectionJobsResponse = await
comprehendClient.ListTopicsDetectionJobsAsync(new
ListTopicsDetectionJobsRequest());
        foreach (var props in
listTopicsDetectionJobsResponse.TopicsDetectionJobPropertiesList)
        {
            PrintJobProperties(props);
        }
    }
}
```

```
        }

    ///<summary>
    /// This method is a helper method that displays the job properties
    /// from the call to StartTopicsDetectionJobRequest.
    ///</summary>
    ///<param name="props">A list of properties from the call to
    /// StartTopicsDetectionJobRequest.</param>
    private static void PrintJobProperties(TopicsDetectionJobProperties props)
    {
        Console.WriteLine($"JobId: {props.JobId}, JobName: {props.JobName},
JobStatus: {props.JobStatus}");
        Console.WriteLine($"NumberofTopics: {props.NumberofTopics}\nInputS3Uri:
{props.InputDataConfig.S3Uri}");
        Console.WriteLine($"InputFormat: {props.InputDataConfig.InputFormat},
OutputS3Uri: {props.OutputDataConfig.S3Uri}");
    }
}
```

- For API details, see [StartTopicsDetectionJob](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client

    def start_job(
            self, job_name, input_bucket, input_key, input_format, output_bucket,
            output_key, data_access_role_arn):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
        input bucket and written to the specified output bucket. Output data is
        stored in a tar archive compressed in gzip format. The job runs asynchronously, so
        you can call `describe_topics_detection_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
            bucket. If multiple objects have the same prefix, all
            of them are used.
        :param input_format: The format of the input data, either one document per
            file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
            grants Comprehend permission to read from the
```

```
        input bucket and write to the output bucket.  
    :return: Information about the job, including the job ID.  
    """  
    try:  
        response = self.comprehend_client.start_topics_detection_job(  
            JobName=job_name,  
            DataAccessRoleArn=data_access_role_arn,  
            InputDataConfig={  
                'S3Uri': f's3://{input_bucket}/{input_key}',  
                'InputFormat': input_format.value},  
            OutputDataConfig={'S3Uri': f's3://{output_bucket}/{output_key}'})  
        logger.info("Started topic modeling job %s.", response['JobId'])  
    except ClientError:  
        logger.exception("Couldn't start topic modeling job.")  
        raise  
    else:  
        return response
```

- For API details, see [StartTopicsDetectionJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios for Amazon Comprehend using AWS SDKs

The following code examples show how to use Amazon Comprehend with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Detect document elements with Amazon Comprehend and an AWS SDK \(p. 300\)](#)
- [Run an Amazon Comprehend topic modeling job on sample data using an AWS SDK \(p. 304\)](#)
- [Train a custom Amazon Comprehend classifier and classify documents using an AWS SDK \(p. 306\)](#)

## Detect document elements with Amazon Comprehend and an AWS SDK

The following code example shows how to:

- Detect languages in a document.
- Detect entities in a document.
- Detect key phrases in a document.
- Detect personally identifiable information (PII) in a document.
- Detect the sentiment of a document.
- Detect syntax elements in a document.

### Python

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps Amazon Comprehend actions.

```
import logging
```

```
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client

    def detect_languages(self, text):
        """
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_dominant_language(Text=text)
            languages = response['Languages']
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
            return languages

    def detect_entities(self, text, language_code):
        """
        Detects entities in a document. Entities can be things like people and
        places or other common terms.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_entities(
                Text=text, LanguageCode=language_code)
            entities = response['Entities']
            logger.info("Detected %s entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect entities.")
            raise
        else:
            return entities

    def detect_key_phrases(self, text, language_code):
        """
        Detects key phrases in a document. A key phrase is typically a noun and its
        modifiers.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of key phrases along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_key_phrases(
                Text=text, LanguageCode=language_code)
            phrases = response['KeyPhrases']
        except ClientError:
            logger.exception("Couldn't detect key phrases.")
            raise
        else:
            return phrases
```

```
        logger.info("Detected %s phrases.", len(phrases))
    except ClientError:
        logger.exception("Couldn't detect phrases.")
        raise
    else:
        return phrases

def detect_pii(self, text, language_code):
    """
    Detects personally identifiable information (PII) in a document. PII can be
    things like names, account numbers, or addresses.

    :param text: The document to inspect.
    :param language_code: The language of the document.
    :return: The list of PII entities along with their confidence scores.
    """
    try:
        response = self.comprehend_client.detect_pii_entities(
            Text=text, LanguageCode=language_code)
        entities = response['Entities']
        logger.info("Detected %s PII entities.", len(entities))
    except ClientError:
        logger.exception("Couldn't detect PII entities.")
        raise
    else:
        return entities

def detect_sentiment(self, text, language_code):
    """
    Detects the overall sentiment expressed in a document. Sentiment can
    be positive, negative, neutral, or a mixture.

    :param text: The document to inspect.
    :param language_code: The language of the document.
    :return: The sentiments along with their confidence scores.
    """
    try:
        response = self.comprehend_client.detect_sentiment(
            Text=text, LanguageCode=language_code)
        logger.info("Detected primary sentiment %s.", response['Sentiment'])
    except ClientError:
        logger.exception("Couldn't detect sentiment.")
        raise
    else:
        return response

def detect_syntax(self, text, language_code):
    """
    Detects syntactical elements of a document. Syntax tokens are portions of
    text along with their use as parts of speech, such as nouns, verbs, and
    interjections.

    :param text: The document to inspect.
    :param language_code: The language of the document.
    :return: The list of syntax tokens along with their confidence scores.
    """
    try:
        response = self.comprehend_client.detect_syntax(
            Text=text, LanguageCode=language_code)
        tokens = response['SyntaxTokens']
        logger.info("Detected %s syntax tokens.", len(tokens))
    except ClientError:
        logger.exception("Couldn't detect syntax.")
        raise
    else:
        return tokens
```

Call functions on the wrapper class to detect entities, phrases, and more in a document.

```
def usage_demo():
    print('*'*88)
    print("Welcome to the Amazon Comprehend detection demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    comp_detect = ComprehendDetect(boto3.client('comprehend'))
    with open('detect_sample.txt') as sample_file:
        sample_text = sample_file.read()

    demo_size = 3

    print("Sample text used for this demo:")
    print('*'*88)
    print(sample_text)
    print('*'*88)

    print("Detecting languages.")
    languages = comp_detect.detect_languages(sample_text)
    pprint(languages)
    lang_code = languages[0]['LanguageCode']

    print("Detecting entities.")
    entities = comp_detect.detect_entities(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(entities[:demo_size])

    print("Detecting key phrases.")
    phrases = comp_detect.detect_key_phrases(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(phrases[:demo_size])

    print("Detecting personally identifiable information (PII).")
    pii_entities = comp_detect.detect_pii(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(pii_entities[:demo_size])

    print("Detecting sentiment.")
    sentiment = comp_detect.detect_sentiment(sample_text, lang_code)
    print(f"Sentiment: {sentiment['Sentiment']}")
    print("SentimentScore:")
    pprint(sentiment['SentimentScore'])

    print("Detecting syntax elements.")
    syntax_tokens = comp_detect.detect_syntax(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(syntax_tokens[:demo_size])

    print("Thanks for watching!")
    print('*'*88)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [DetectDominantLanguage](#)
  - [DetectEntities](#)
  - [DetectKeyPhrases](#)
  - [DetectPiiEntities](#)

- [DetectSentiment](#)
- [DetectSyntax](#)

## Run an Amazon Comprehend topic modeling job on sample data using an AWS SDK

The following code example shows how to:

- Run an Amazon Comprehend topic modeling job on sample data.
- Get information about the job.
- Extract job output data from Amazon Simple Storage Service (Amazon S3).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a wrapper class to call Amazon Comprehend topic modeling actions.

```
class ComprehendTopicModeler:  
    """Encapsulates a Comprehend topic modeler."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
  
    def start_job(  
        self, job_name, input_bucket, input_key, input_format, output_bucket,  
        output_key, data_access_role_arn):  
        """  
        Starts a topic modeling job. Input is read from the specified Amazon S3  
        input bucket and written to the specified output bucket. Output data is  
        stored  
        in a tar archive compressed in gzip format. The job runs asynchronously, so  
        you  
        can call `describe_topics_detection_job` to get job status until it  
        returns a status of SUCCEEDED.  
  
        :param job_name: The name of the job.  
        :param input_bucket: An Amazon S3 bucket that contains job input.  
        :param input_key: The prefix used to find input data in the input  
            bucket. If multiple objects have the same prefix, all  
            of them are used.  
        :param input_format: The format of the input data, either one document per  
            file or one document per line.  
        :param output_bucket: The Amazon S3 bucket where output data is written.  
        :param output_key: The prefix prepended to the output data.  
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that  
            grants Comprehend permission to read from the  
            input bucket and write to the output bucket.  
        :return: Information about the job, including the job ID.  
        """  
        try:  
            response = self.comprehend_client.start_topics_detection_job(  
                JobName=job_name,  
                DataAccessRoleArn=data_access_role_arn,
```

```

InputDataConfig={
    'S3Uri': f's3://{input_bucket}/{input_key}',
    'InputFormat': input_format.value,
    'OutputDataConfig': {'S3Uri': f's3://{output_bucket}/{output_key}'})
logger.info("Started topic modeling job %s.", response['JobId'])
except ClientError:
    logger.exception("Couldn't start topic modeling job.")
    raise
else:
    return response

def describe_job(self, job_id):
    """
    Gets metadata about a topic modeling job.

    :param job_id: The ID of the job to look up.
    :return: Metadata about the job.
    """
    try:
        response = self.comprehend_client.describe_topics_detection_job(
            JobId=job_id)
        job = response['TopicsDetectionJobProperties']
        logger.info("Got topic detection job %s.", job_id)
    except ClientError:
        logger.exception("Couldn't get topic detection job %s.", job_id)
        raise
    else:
        return job

def list_jobs(self):
    """
    Lists topic modeling jobs for the current account.

    :return: The list of jobs.
    """
    try:
        response = self.comprehend_client.list_topics_detection_jobs()
        jobs = response['TopicsDetectionJobPropertiesList']
        logger.info("Got %s topic detection jobs.", len(jobs))
    except ClientError:
        logger.exception("Couldn't get topic detection jobs.")
        raise
    else:
        return jobs

```

Use the wrapper class to run a topic modeling job and get job data.

```

def usage_demo():
    print('*'*88)
    print("Welcome to the Amazon Comprehend topic modeling demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    input_prefix = 'input/'
    output_prefix = 'output/'
    demo_resources = ComprehendDemoResources(
        boto3.resource('s3'), boto3.resource('iam'))
    topic_modeler = ComprehendTopicModeler(boto3.client('comprehend'))

    print("Setting up storage and security resources needed for the demo.")
    demo_resources.setup('comprehend-topic-modeler-demo')
    print("Copying sample data from public bucket into input bucket.")
    demo_resources.bucket.copy()

```

```
{'Bucket': 'public-sample-us-west-2', 'Key': 'TopicModeling/Sample.txt'},  
f'{input_prefix}sample.txt')  
  
print("Starting topic modeling job on sample data.")  
job_info = topic_modeler.start_job(  
    'demo-topic-modeling-job', demo_resources.bucket.name, input_prefix,  
    JobInputFormat.per_line, demo_resources.bucket.name, output_prefix,  
    demo_resources.data_access_role.arn)  
  
print(f"Waiting for job {job_info['JobId']} to complete. This typically takes "  
     f"20 - 30 minutes.")  
job_waiter = JobCompleteWaiter(topic_modeler.comprehend_client)  
job_waiter.wait(job_info['JobId'])  
  
job = topic_modeler.describe_job(job_info['JobId'])  
print(f"Job {job['JobId']} complete:")  
pprint(job)  
  
print(f"Getting job output data from the output Amazon S3 bucket: "  
     f"{job['OutputDataConfig']['S3Uri']}")  
job_output = demo_resources.extract_job_output(job)  
lines = 10  
print(f"First {lines} lines of document topics output:")  
pprint(job_output['doc-topics.csv']['data'][:lines])  
print(f"First {lines} lines of terms output:")  
pprint(job_output['topic-terms.csv']['data'][:lines])  
  
print("Cleaning up resources created for the demo.")  
demo_resources.cleanup()  
  
print("Thanks for watching!")  
print(' - *88)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [DescribeTopicsDetectionJob](#)
  - [ListTopicsDetectionJobs](#)
  - [StartTopicsDetectionJob](#)

## Train a custom Amazon Comprehend classifier and classify documents using an AWS SDK

The following code example shows how to:

- Create an Amazon Comprehend multi-label classifier.
- Train the classifier on sample data.
- Run a classification job on a second set of data.
- Extract the job output data from Amazon Simple Storage Service (Amazon S3).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a wrapper class to call Amazon Comprehend document classifier actions.

```

class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None

    def create(
            self, name, language_code, training_bucket, training_key,
            data_access_role_arn, mode):
        """
        Creates a custom classifier. After the classifier is created, it
        immediately
        starts training on the data found in the specified Amazon S3 bucket.
        Training
        can take 30 minutes or longer. The `describe_document_classifier` function
        can be used to get training status and returns a status of TRAINED when the
        classifier is ready to use.

        :param name: The name of the classifier.
        :param language_code: The language the classifier can operate on.
        :param training_bucket: The Amazon S3 bucket that contains the training
        data.
        :param training_key: The prefix used to find training data in the training
        bucket. If multiple objects have the same prefix, all
        of them are used.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
        grants Comprehend permission to read from the
        training bucket.
        :return: The ARN of the newly created classifier.
        """
        try:
            response = self.comprehend_client.create_document_classifier(
                DocumentClassifierName=name,
                LanguageCode=language_code,
                InputDataConfig={'S3Uri': f's3://{training_bucket}/'
                {training_key}'},
                DataAccessRoleArn=data_access_role_arn,
                Mode=mode.value)
            self.classifier_arn = response['DocumentClassifierArn']
            logger.info("Started classifier creation. Arn is: %s.", self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't create classifier %s.", name)
            raise
        else:
            return self.classifier_arn

    def describe(self, classifier_arn=None):
        """
        Gets metadata about a custom classifier, including its current status.

        :param classifier_arn: The ARN of the classifier to look up.
        :return: Metadata about the classifier.
        """
        if classifier_arn is not None:
            self.classifier_arn = classifier_arn
        try:
            response = self.comprehend_client.describe_document_classifier(
                DocumentClassifierArn=self.classifier_arn)
            classifier = response['DocumentClassifierProperties']
            logger.info("Got classifier %s.", self.classifier_arn)
        except ClientError:

```

```
        logger.exception("Couldn't get classifier %s.", self.classifier_arn)
        raise
    else:
        return classifier

def list(self):
    """
    Lists custom classifiers for the current account.

    :return: The list of classifiers.
    """
    try:
        response = self.comprehend_client.list_document_classifiers()
        classifiers = response['DocumentClassifierPropertiesList']
        logger.info("Got %s classifiers.", len(classifiers))
    except ClientError:
        logger.exception("Couldn't get classifiers.", )
        raise
    else:
        return classifiers

def delete(self):
    """
    Deletes the classifier.

    """
    try:
        self.comprehend_client.delete_document_classifier(
            DocumentClassifierArn=self.classifier_arn)
        logger.info("Deleted classifier %s.", self.classifier_arn)
        self.classifier_arn = None
    except ClientError:
        logger.exception("Couldn't deleted classifier %s.",
                         self.classifier_arn)
        raise

def start_job(
    self, job_name, input_bucket, input_key, input_format, output_bucket,
    output_key, data_access_role_arn):
    """
    Starts a classification job. The classifier must be trained or the job
    will fail. Input is read from the specified Amazon S3 input bucket and
    written to the specified output bucket. Output data is stored in a tar
    archive compressed in gzip format. The job runs asynchronously, so you can
    call `describe_document_classification_job` to get job status until it
    returns a status of SUCCEEDED.

    :param job_name: The name of the job.
    :param input_bucket: The Amazon S3 bucket that contains input data.
    :param input_key: The prefix used to find input data in the input
                      bucket. If multiple objects have the same prefix, all
                      of them are used.
    :param input_format: The format of the input data, either one document per
                        file or one document per line.
    :param output_bucket: The Amazon S3 bucket where output data is written.
    :param output_key: The prefix prepended to the output data.
    :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                grants Comprehend permission to read from the
                                input bucket and write to the output bucket.
    :return: Information about the job, including the job ID.
    """
    try:
        response = self.comprehend_client.start_document_classification_job(
            DocumentClassifierArn=self.classifier_arn,
            JobName=job_name,
            InputDataConfig={
                'S3Uri': f's3://{input_bucket}/{input_key}',
```

```

        'InputFormat': input_format.value},
        OutputDataConfig={'S3Uri': f's3://{output_bucket}/{output_key}'},
        DataAccessRoleArn=data_access_role_arn)
    logger.info(
        "Document classification job %s is %s.", job_name,
        response['JobStatus'])
except ClientError:
    logger.exception("Couldn't start classification job %s.", job_name)
    raise
else:
    return response

def describe_job(self, job_id):
    """
    Gets metadata about a classification job.

    :param job_id: The ID of the job to look up.
    :return: Metadata about the job.
    """
    try:
        response = self.comprehend_client.describe_document_classification_job(
            JobId=job_id)
        job = response['DocumentClassificationJobProperties']
        logger.info("Got classification job %s.", job['JobName'])
    except ClientError:
        logger.exception("Couldn't get classification job %s.", job_id)
        raise
    else:
        return job

def list_jobs(self):
    """
    Lists the classification jobs for the current account.

    :return: The list of jobs.
    """
    try:
        response = self.comprehend_client.list_document_classification_jobs()
        jobs = response['DocumentClassificationJobPropertiesList']
        logger.info("Got %s document classification jobs.", len(jobs))
    except ClientError:
        logger.exception("Couldn't get document classification jobs.", )
        raise
    else:
        return jobs

```

Create a class to help run the scenario.

```

class ClassifierDemo:
    """
    Encapsulates functions used to run the demonstration.
    """
    def __init__(self, demo_resources):
        """
        :param demo_resources: A ComprehendDemoResources class that manages
        resources
                           for the demonstration.
        """
        self.demo_resources = demo_resources
        self.training_prefix = 'training/'
        self.input_prefix = 'input/'
        self.input_format = JobInputFormat.per_line
        self.output_prefix = 'output/'

```

```
def setup(self):
    """Creates AWS resources used by the demo."""
    self.demo_resources.setup('comprehend-classifier-demo')

def cleanup(self):
    """Deletes AWS resources used by the demo."""
    self.demo_resources.cleanup()

@staticmethod
def _sanitize_text(text):
    """Removes characters that cause errors for the document parser."""
    return text.replace('\r', ' ').replace('\n', ' ').replace(',', ';')

@staticmethod
def _get_issues(query, issue_count):
    """
    Gets issues from GitHub using the specified query parameters.

    :param query: The query string used to request issues from the GitHub API.
    :param issue_count: The number of issues to retrieve.
    :return: The list of issues retrieved from GitHub.
    """

    issues = []
    logger.info("Requesting issues from %s?%s.", GITHUB_SEARCH_URL, query)
    response = requests.get(
        f'{GITHUB_SEARCH_URL}?{query}&per_page={issue_count}')
    if response.status_code == 200:
        issue_page = response.json()['items']
        logger.info("Got %s issues.", len(issue_page))
        issues = [
            {
                'title': ClassifierDemo._sanitize_text(issue['title']),
                'body': ClassifierDemo._sanitize_text(issue['body']),
                'labels': {label['name'] for label in issue['labels']}
            } for issue in issue_page]
    else:
        logger.error(
            "GitHub returned error code %s with message %s.",
            response.status_code, response.json())
    logger.info("Found %s issues.", len(issues))
    return issues

def get_training_issues(self, training_labels):
    """
    Gets issues used for training the custom classifier. Training issues are
    closed issues from the Boto3 repo that have known labels. Comprehend
    requires a minimum of ten training issues per label.

    :param training_labels: The issue labels to use for training.
    :return: The set of issues used for training.
    """

    issues = []
    per_label_count = 15
    for label in training_labels:
        issues += self._get_issues(
            f'q=type:issue+repo:boto/boto3+state:closed+label:{label}',
            per_label_count)
    for issue in issues:
        issue['labels'] = issue['labels'].intersection(training_labels)
    return issues

def get_input_issues(self, training_labels):
    """
    Gets input issues from GitHub. For demonstration purposes, input issues
    are open issues from the Boto3 repo with known labels, though in practice
    any issue could be submitted to the classifier for labeling.
    """
```

```

:param training_labels: The set of labels to query for.
:return: The set of issues used for input.
"""
issues = []
per_label_count = 5
for label in training_labels:
    issues += self._get_issues(
        f'q=type:issue+repo:boto/boto3+state:open+label:{label}',
        per_label_count)
return issues

def upload_issue_data(self, issues, training=False):
    """
    Uploads issue data to an Amazon S3 bucket, either for training or for
    input.
    The data is first put into the format expected by Comprehend. For training,
    the set of pipe-delimited labels is prepended to each document. For
    input, labels are not sent.

    :param issues: The set of issues to upload to Amazon S3.
    :param training: Indicates whether the issue data is used for training or
                     input.
    """
    try:
        obj_key = (
            self.training_prefix if training else self.input_prefix) +
        'issues.txt'
        if training:
            issue_strings = [
                f"{'|'.join(issue['labels'])},{issue['title']} {issue['body']}"
                for issue in issues]
        else:
            issue_strings = [
                f"{issue['title']} {issue['body']}" for issue in issues]
        issue_bytes = BytesIO('\n'.join(issue_strings).encode('utf-8'))
        self.demo_resources.bucket.upload_fileobj(issue_bytes, obj_key)
        logger.info(
            "Uploaded data as %s to bucket %s.", obj_key,
            self.demo_resources.bucket.name)
    except ClientError:
        logger.exception(
            "Couldn't upload data to bucket %s.",
            self.demo_resources.bucket.name)
        raise

    def extract_job_output(self, job):
        """Extracts job output from Amazon S3."""
        return self.demo_resources.extract_job_output(job)

    @staticmethod
    def reconcile_job_output(input_issues, output_dict):
        """
        Reconciles job output with the list of input issues. Because the input
        issues
        have known labels, these can be compared with the labels added by the
        classifier to judge the accuracy of the output.

        :param input_issues: The list of issues used as input.
        :param output_dict: The dictionary of data that is output by the
                           classifier.
        :return: The list of reconciled input and output data.
        """
        reconciled = []
        for archive in output_dict.values():
            for line in archive['data']:
                in_line = int(line['Line'])

```

```
in_labels = input_issues[in_line]['labels']
out_labels = {label['Name'] for label in line['Labels']
              if float(label['Score']) > 0.3}
reconciled.append(
    f"{line['File']}, line {in_line} has labels {in_labels}.\\n"
    f"\tClassifier assigned {out_labels}.")
logger.info("Reconciled input and output labels.")
return reconciled
```

Train a classifier on a set of GitHub issues with known labels, then send a second set of GitHub issues to the classifier so that they can be labeled.

```
def usage_demo():
    print('*'*88)
    print("Welcome to the Amazon Comprehend custom document classifier demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    comp_demo = ClassifierDemo(ComprehendDemoResources(
        boto3.resource('s3'), boto3.resource('iam')))
    comp_classifier = ComprehendClassifier(boto3.client('comprehend'))
    classifier_trained_waiter = ClassifierTrainedWaiter(
        comp_classifier.comprehend_client)
    training_labels = {'bug', 'feature-request', 'dynamodb', 's3'}

    print("Setting up storage and security resources needed for the demo.")
    comp_demo.setup()

    print("Getting training data from GitHub and uploading it to Amazon S3.")
    training_issues = comp_demo.get_training_issues(training_labels)
    comp_demo.upload_issue_data(training_issues, True)

    classifier_name = 'doc-example-classifier'
    print(f"Creating document classifier {classifier_name}.")
    comp_classifier.create(
        classifier_name, 'en',
        comp_demo.demo_resources.bucket.name,
        comp_demo.training_prefix,
        comp_demo.demo_resources.data_access_role.arn,
        ClassifierMode.multi_label)
    print(f"Waiting until {classifier_name} is trained. This typically takes "
          f"30-40 minutes.")
    classifier_trained_waiter.wait(comp_classifier.classifier_arn)

    print(f"Classifier {classifier_name} is trained:")
    pprint(comp_classifier.describe())

    print("Getting input data from GitHub and uploading it to Amazon S3.")
    input_issues = comp_demo.get_input_issues(training_labels)
    comp_demo.upload_issue_data(input_issues)

    print("Starting classification job on input data.")
    job_info = comp_classifier.start_job(
        'issue_classification_job',
        comp_demo.demo_resources.bucket.name,
        comp_demo.input_prefix,
        comp_demo.input_format,
        comp_demo.demo_resources.bucket.name,
        comp_demo.output_prefix,
        comp_demo.demo_resources.data_access_role.arn)
    print(f"Waiting for job {job_info['JobId']} to complete.")
    job_waiter = JobCompleteWaiter(comp_classifier.comprehend_client)
```

```
job_waiter.wait(job_info['JobId'])

job = comp_classifier.describe_job(job_info['JobId'])
print(f"Job {job['JobId']} complete:")
pprint(job)

print(f"Getting job output data from Amazon S3: "
      f"{job['OutputDataConfig']['S3Uri']}")
job_output = comp_demo.extract_job_output(job)
print("Job output:")
pprint(job_output)

print("Reconciling job output with labels from GitHub:")
reconciled_output = comp_demo.reconcile_job_output(input_issues, job_output)
print(*reconciled_output, sep='\n')

answer = input(f"Do you want to delete the classifier {classifier_name} (y/n)?")
if answer.lower() == 'y':
    print(f"Deleting {classifier_name}.")
    comp_classifier.delete()

print("Cleaning up resources created for the demo.")
comp_demo.cleanup()

print("Thanks for watching!")
print('*'*88)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateDocumentClassifier](#)
  - [DeleteDocumentClassifier](#)
  - [DescribeDocumentClassificationJob](#)
  - [DescribeDocumentClassifier](#)
  - [ListDocumentClassificationJobs](#)
  - [ListDocumentClassifiers](#)
  - [StartDocumentClassificationJob](#)

## Cross-service examples for Amazon Comprehend using AWS SDKs

The following code examples show how to use Amazon Comprehend with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Build an Amazon Transcribe streaming app \(p. 313\)](#)
- [Create an Amazon Lex Chatbot within a web application to engage your web site visitors \(p. 314\)](#)
- [Create a web application that sends and retrieves messages by using the Amazon Simple Queue Service API \(p. 315\)](#)
- [Detect entities in text extracted from an image using an AWS SDK \(p. 315\)](#)

### Build an Amazon Transcribe streaming app

The following code example shows how to build an app that records, transcribes, and translates live audio in real-time, and emails the results.

JavaScript

### SDK for JavaScript V3

Shows how to use Amazon Transcribe to build an app that records, transcribes, and translates live audio in real-time, and emails the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

## Create an Amazon Lex Chatbot within a web application to engage your web site visitors

The following code examples show how to create a Chatbot to engage your web site visitors.

Java

### SDK for Java 2.x

Shows how to use the Amazon Lex API to create a Chatbot within a web application to engage your web site visitors.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

JavaScript

### SDK for JavaScript V3

Shows how to use the Amazon Lex API to create a Chatbot within a web application to engage your web site visitors.

For complete source code and instructions on how to set up and run, see the full example [Building an Amazon Lex chatbot](#) in the AWS SDK for JavaScript developer guide.

#### Services used in this example

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

## Create a web application that sends and retrieves messages by using the Amazon Simple Queue Service API

The following code examples show how to create a messaging application by using the Amazon Simple Queue Service API.

Java

### SDK for Java 2.x

Shows how to use the Amazon Simple Queue Service API to develop a Spring REST API that sends and retrieves messages.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon SQS

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to use the Amazon Simple Queue Service API to develop a Spring REST API that sends and retrieves messages.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon SQS

## Detect entities in text extracted from an image using an AWS SDK

The following code example shows how to use Amazon Comprehend to detect entities in text extracted by Amazon Textract from an image that is stored in Amazon S3.

Python

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) in a Jupyter notebook to detect entities in text that is extracted from an image. This example uses Amazon Textract to extract text from an image stored in Amazon Simple Storage Service (Amazon S3) and Amazon Comprehend to detect entities in the extracted text.

This example is a Jupyter notebook and must be run in an environment that can host notebooks. For instructions on how to run the example using Amazon SageMaker, see the directions in [TextractAndComprehendNotebook.ipynb](#).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon S3
- Amazon Textract

## Code examples for AWS Config using AWS SDKs

The following code examples show how to use AWS Config with an AWS software development kit (SDK).

#### Code examples

- [Actions for AWS Config using AWS SDKs \(p. 316\)](#)
  - [Delete an AWS Config rule using an AWS SDK \(p. 316\)](#)
  - [Describe AWS Config rules using an AWS SDK \(p. 317\)](#)
  - [Put an AWS Config rule using an AWS SDK \(p. 317\)](#)

## Actions for AWS Config using AWS SDKs

The following code examples show how to use AWS Config with AWS SDKs. Each example calls an individual service function.

#### Examples

- [Delete an AWS Config rule using an AWS SDK \(p. 316\)](#)
- [Describe AWS Config rules using an AWS SDK \(p. 317\)](#)
- [Put an AWS Config rule using an AWS SDK \(p. 317\)](#)

## Delete an AWS Config rule using an AWS SDK

The following code example shows how to delete an AWS Config rule.

Python

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ConfigWrapper:  
    """  
    Encapsulates AWS Config functions.  
    """  
    def __init__(self, config_client):  
        """  
        :param config_client: A Boto3 AWS Config client.  
        """  
        self.config_client = config_client  
  
    def delete_config_rule(self, rule_name):  
        """
```

```
Delete the specified rule.

:param rule_name: The name of the rule to delete.
"""
try:
    self.config_client.delete_config_rule(ConfigRuleName=rule_name)
    logger.info("Deleted rule %s.", rule_name)
except ClientError:
    logger.exception("Couldn't delete rule %s.", rule_name)
    raise
```

- For API details, see [DeleteConfigRule](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe AWS Config rules using an AWS SDK

The following code example shows how to describe AWS Config rules.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ConfigWrapper:
    """
    Encapsulates AWS Config functions.
    """
    def __init__(self, config_client):
        """
        :param config_client: A Boto3 AWS Config client.
        """
        self.config_client = config_client

    def describe_config_rule(self, rule_name):
        """
        Gets data for the specified rule.

        :param rule_name: The name of the rule to retrieve.
        :return: The rule data.
        """
        try:
            response = self.config_client.describe_config_rules(
                ConfigRuleNames=[rule_name])
            rule = response['ConfigRules']
            logger.info("Got data for rule %s.", rule_name)
        except ClientError:
            logger.exception("Couldn't get data for rule %s.", rule_name)
            raise
        else:
            return rule
```

- For API details, see [DescribeConfigRules](#) in *AWS SDK for Python (Boto3) API Reference*.

## Put an AWS Config rule using an AWS SDK

The following code example shows how to put an AWS Config rule.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ConfigWrapper:  
    """  
    Encapsulates AWS Config functions.  
    """  
    def __init__(self, config_client):  
        """  
        :param config_client: A Boto3 AWS Config client.  
        """  
        self.config_client = config_client  
  
    def put_config_rule(self, rule_name):  
        """  
        Sets a configuration rule that prohibits making Amazon S3 buckets publicly  
        readable.  
        :param rule_name: The name to give the rule.  
        """  
        try:  
            self.config_client.put_config_rule(  
                ConfigRule={  
                    'ConfigRuleName': rule_name,  
                    'Description': 'S3 Public Read Prohibited Bucket Rule',  
                    'Scope': {  
                        'ComplianceResourceTypes': [  
                            'AWS::S3::Bucket',  
                        ],  
                    },  
                    'Source': {  
                        'Owner': 'AWS',  
                        'SourceIdentifier': 'S3_BUCKET_PUBLIC_READ_PROHIBITED',  
                    },  
                    'InputParameters': '{}',  
                    'ConfigRuleState': 'ACTIVE'  
                }  
            )  
            logger.info("Created configuration rule %s.", rule_name)  
        except ClientError:  
            logger.exception("Couldn't create configuration rule %s.", rule_name)  
            raise
```

- For API details, see [PutConfigRule](#) in *AWS SDK for Python (Boto3) API Reference*.

## Code examples for Device Farm using AWS SDKs

The following code examples show how to use AWS Device Farm with an AWS software development kit (SDK).

### Code examples

- [Scenarios for Device Farm using AWS SDKs \(p. 319\)](#)
  - Run browser tests with Device Farm and take screenshots using an AWS SDK (p. 319)
  - Upload and test mobile device packages with Device Farm using an AWS SDK (p. 322)

## Scenarios for Device Farm using AWS SDKs

The following code examples show how to use AWS Device Farm with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Run browser tests with Device Farm and take screenshots using an AWS SDK \(p. 319\)](#)
- [Upload and test mobile device packages with Device Farm using an AWS SDK \(p. 322\)](#)

## Run browser tests with Device Farm and take screenshots using an AWS SDK

The following code example shows how to run browser tests with Device Farm and take screenshots.

### Python

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use PyTest and Selenium to browse to specified websites, take screenshots, and compare actual website content with expected content.

```
import datetime
import os
import subprocess
import boto3
import pytest
from selenium import webdriver
from selenium.webdriver import DesiredCapabilities
from selenium.webdriver.common import by
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait

def get_git_hash():
    """
    Get the short Git hash of the current commit of the repository
    """
    try:
        return subprocess.check_output(
            ['git', 'rev-parse', '--short', 'HEAD']).decode('utf-8').strip()
    except:
        return "norepo"

class TestHelloSuite:
    """
    Our test suite.

    This style of test suite allows us to use setup_method and teardown_method.
    """
    def save_screenshot(self, name):
        self.driver.save_screenshot(os.path.join(self.screenshot_path, name))
```

```
def setup_method(self, method):
    """
    Set up a test.

    This makes sure that the session for an individual test is ready.

    The AWS credentials are read from the default ~/.aws/credentials or from
    the
        command line by setting the AWS_ACCESS_KEY_ID and AWS_SECRET_KEY
    environment
        variables.

    The project Amazon Resource Name (ARN) is determined by the PROJECT_ARN
    environment variable.
    """
    devicefarm_client = boto3.client("devicefarm")
    project_arn = os.environ.get("PROJECT_ARN", None)
    if project_arn is None:
        raise ValueError("Must set PROJECT_ARN")
    # Request a driver hub URL for the Selenium client
    testgrid_url_response = devicefarm_client.create_test_grid_url(
        projectArn=project_arn,
        expiresInSeconds=300)

    # We want a directory to save our files into. We're going to make a
    directory
        # in the current directory that holds our results.
        self.screenshot_path = os.path.join(
            '.', 'results', get_git_hash() + '-' +
            (datetime.date.today().isoformat()))
        if not os.path.exists(self.screenshot_path):
            os.makedirs(self.screenshot_path, exist_ok=True)

    # We want a Firefox instance on Windows
    desired_cap = DesiredCapabilities.FIREFOX
    desired_cap['platform'] = 'windows'
    desired_cap['BrowserVersion'] = 'latest'

    # Configure the webdriver with the appropriate remote endpoint.
    self.driver = webdriver.Remote(testgrid_url_response['url'], desired_cap)

    #
    # Auto-Tagging
    #

    # In order to get the Session ARN, we need to look up the session by the
    # Project ARN and session ID (from the driver).
    testgrid_session_arn_response = devicefarm_client.get_test_grid_session(
        projectArn=project_arn,
        sessionId=self.driver.session_id
    )

    # Save the session's ARN so we can tag the session.
    self.session_arn = testgrid_session_arn_response['testGridSession']['arn']

    # In order to tag it, we're going to use the resourcegroupstaggingapi
    client to
        # add a tag to the session ARN that we just got.
        tag_client = boto3.client('resourcegroupstaggingapi')
        tag_client.tag_resources(
            ResourceARNList=[self.session_arn],
            Tags={
                "TestSuite": f"testsuite {method.__name__}",
                "GitId": get_git_hash()})
    
```

```
def teardown_method(self, method):
```

```

"""
Clean up resources used by each method.
"""

# End the Selenium session so we're off the clock.
self.driver.quit()

@pytest.mark.parametrize(
    'query,leading', [
        pytest.param('Seattle', 'Seattle (/si#ætəl/ (listen) see-AT-əl) is a
seaport city on the West Coast of the United States.'),
        pytest.param('Selenium', "Selenium is a chemical element with the
symbol Se and atomic number 34."),
        pytest.param('Amazon Locker', 'Amazon Locker is a self-service package
delivery service offered by online retailer Amazon.'),
        pytest.param('Kootenai Falls','Kootenai Falls is a waterfall on the
Kootenay River located in Lincoln County, Montana, just off U.S. Route 2.'),
        pytest.param('Dorayaki', 'Dorayaki (####, ####, ####, ####) is a type
of Japanese confection.'),
        pytest.param('Robot Face', '<|°_°|> (also known as Robot Face or
Robot)')
    ])
def test_first_paragraph_text(self, query, leading):
    """
    This test looks at the first paragraph of a page on Wikipedia, comparing it
    to
    a known leading sentence.

    If the leading sentence matches, the test passes. A screenshot is taken
    before
    the final assertion is made, letting us debug if something isn't right.
    """

    # Open the main page of Wikipedia
    self.driver.get("https://en.wikipedia.org/wiki/Main_Page")
    # Find the search box, enter a query, and press enter
    search_input = self.driver.find_element(By.ID, "searchInput")
    search_input.click()
    search_input.send_keys(query)
    search_input.send_keys(Keys.ENTER)
    # Wait for the search box to go stale -- This means we've navigated fully.
    WebDriverWait(self.driver,
5).until(expected_conditions.staleness_of(search_input))
    # Get the leading paragraph of the article.
    lead = leading.lower()
    # Find the element...
    lead_para = self.driver.find_element(
        By.XPATH, "//div[@class='mw-parser-output']//p[not(@class)]")
    # ... and copy out its text.
    our_text = lead_para.text.lower()
    our_text = our_text[:len(lead)]
    # Take a screenshot and compare the strings.
    self.save_screenshot(f"leadingpara_{query}.png")
    assert our_text.startswith(lead)

@pytest.mark.parametrize(
    "query,expected", [
        pytest.param("Automation Testing", "Test Automation"),
        pytest.param("DevOps", "DevOps"),
        pytest.param("Jackdaws Love My Big Sphinx Of Quartz", "Pangram"),
        pytest.param("EarthBound", "EarthBound"),
        pytest.param("Covered Bridges Today", "Covered Bridges Today"),
        pytest.param("Kurt Godel", "Kurt Gödel"),
        pytest.param("N//ng language", "N#ng language"),
        pytest.param("Who the Fuck Is Jackson Pollock?", "Who the $&% Is
Jackson Pollock?")
    ])
def test_redirect_titles(self, query, expected):

```

```
"""
A test comparing pages we expect to (or not to) redirect on Wikipedia.

This test checks to see that the page ("query") redirects (or doesn't) to
the
"expected" page title. Several of these are common synonyms ("Jackdaws...")
while others are because of characters untypable by most keyboards ("N#ng
language")

A screenshot is taken just before the final assertion is made to aid in
debugging and verification.
"""

# Open the main page of Wikipedia
self.driver.get("https://en.wikipedia.org/wiki/Main_Page")
# Find the search box, enter some text into it, and send an enter key.
search_input = self.driver.find_element(By.ID, "searchInput")
search_input.click()
search_input.send_keys(query)
search_input.send_keys(Keys.ENTER)
# wait until the page has rolled over -- once the search input handle is
stale,
# the browser has navigated.
WebDriverWait(self.driver,
5).until(expected_conditions.staleness_of(search_input))
# Get the first heading & take a screenshot
our_text = self.driver.find_element(By.ID, "firstHeading").text.lower()
self.save_screenshot(f"redirect_{query}.png")
# did it match?
assert our_text == expected.lower()
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateTestGridUrl](#)
  - [GetTestGridSession](#)

## Upload and test mobile device packages with Device Farm using an AWS SDK

The following code example shows how to upload and test mobile device packages with Device Farm.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Upload compiled Android application and test packages to Device Farm, start a test, wait for test completion, and report the results.

```
import boto3
import os
import requests
import string
import random
import datetime
import time

# Update this dict with your own values before you run the example:
config = {
```

```

# This is our app under test.
"appFilePath": "app-debug.apk",
"projectArn": "arn:aws:devicefarm:us-west-2:111222333444:project:581f5703-
e040-4ac9-b7ae-0ba007bfb8e6",
# Since we care about the most popular devices, we'll use a curated pool.
"testSpecArn": "arn:aws:devicefarm:us-west-2::upload:20fcf771-eae3-4137-
aa76-92e17fb3131b",
"poolArn": "arn:aws:devicefarm:us-
west-2::devicepool:4a869d91-6f17-491f-9a95-0a601aee2406",
"namePrefix": "MyAppTest",
# This is our test package. This tutorial won't go into how to make these.
"testPackage": "tests.zip"
}

client = boto3.client('devicefarm')

unique =
config['namePrefix']+ "-" +(datetime.date.today().isoformat())+''.join(random.sample(string.ascii
8))

print(f"The unique identifier for this run is '{unique}'. All uploads will be
prefixed "
      f"with this.")

def upload_df_file(filename, type_, mime='application/octet-stream'):
    upload_response = client.create_upload(
        projectArn=config['projectArn'],
        name=unique+"_"+os.path.basename(filename),
        type=type_,
        contentType=mime)
    upload_arn = upload_response['upload']['arn']
    # Extract the URL of the upload and use Requests to upload it.
    upload_url = upload_response['upload']['url']
    with open(filename, 'rb') as file_stream:
        print(f"Uploading {filename} to Device Farm as "
              f"{upload_response['upload']['name']}... ", end='')
        put_req = requests.put(
            upload_url, data=file_stream, headers={"content-type": mime})
        print(' done')
        if not put_req.ok:
            raise Exception(f"Couldn't upload. Requests says: {put_req.reason}")
    started = datetime.datetime.now()
    while True:
        print(f"Upload of {filename} in state {upload_response['upload']['status']}"
        " "
              f"after "+str(datetime.datetime.now() - started))
        if upload_response['upload']['status'] == 'FAILED':
            raise Exception(
                f"The upload failed processing. Device Farm says the reason is: \n"
                f"{{+upload_response['upload']['message']}}")
        if upload_response['upload']['status'] == 'SUCCEEDED':
            break
        time.sleep(5)
        upload_response = client.get_upload(arn=upload_arn)
    print("")
    return upload_arn

our_upload_arn = upload_df_file(config['appFilePath'], "ANDROID_APP")
our_test_package_arn = upload_df_file(config['testPackage'],
'APPIUM_PYTHON_TEST_PACKAGE')
print(our_upload_arn, our_test_package_arn)

response = client.schedule_run(
    projectArn=config["projectArn"],

```

```

appArn=our_upload_arn,
devicePoolArn=config["poolArn"],
name=unique,
test={
    "type" :"APPIUM_PYTHON",
    "testSpecArn": config["testSpecArn"],
    "testPackageArn": our_test_package_arn})
run_arn = response['run']['arn']
start_time = datetime.datetime.now()
print(f"Run {unique} is scheduled as arn {run_arn} ")

state = 'UNKNOWN'
try:
    while True:
        response = client.get_run(arn=run_arn)
        state = response['run']['status']
        if state == 'COMPLETED' or state == 'ERRORED':
            break
        else:
            print(f" Run {unique} in state {state}, total "
                  f"time {datetime.datetime.now() - start_time}")
            time.sleep(10)
except:
    client.stop_run(arn=run_arn)
    exit(1)

print(f"Tests finished in state {state} after {datetime.datetime.now() - start_time}")
# Pull all the logs.
jobs_response = client.list_jobs(arn=run_arn)
# Save the output somewhere, using the unique value.
save_path = os.path.join(os.getcwd(), 'results', unique)
os.mkdir(save_path)
# Save the last run information.
for job in jobs_response['jobs']:
    job_name = job['name']
    os.makedirs(os.path.join(save_path, job_name), exist_ok=True)
    # Get each suite within the job.
    suites = client.list_suites(arn=job['arn'])['suites']
    for suite in suites:
        for test in client.list_tests(arn=suite['arn'])['tests']:
            # Get the artifacts.
            for artifact_type in ['FILE', 'SCREENSHOT', 'LOG']:
                artifacts = client.list_artifacts(
                    type=artifact_type, arn=test['arn'])['artifacts']
                for artifact in artifacts:
                    # Replace `:` because it has a special meaning in Windows &
macOS.
                    path_to = os.path.join(
                        save_path, job_name, suite['name'],
test['name'].replace(':', '_'))
                    os.makedirs(path_to, exist_ok=True)
                    filename =
artifac
artifac['type']+ "_" + artifact['name'] + "." + artifact['extension']
                    artifact_save_path = os.path.join(path_to, filename)
                    print(f"Downloading {artifact_save_path}")
                    with open(artifact_save_path, 'wb') as fn:
                        with requests.get(artifact['url'], allow_redirects=True) as
request:
                            fn.write(request.content)
print("Finished")

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateUpload](#)

- [GetRun](#)
- [GetUpload](#)
- [ListArtifacts](#)
- [ListJobs](#)
- [ListSuites](#)
- [ListTests](#)
- [ScheduleRun](#)
- [StopRun](#)

## Code examples for DynamoDB using AWS SDKs

The following code examples show how to use Amazon DynamoDB with an AWS software development kit (SDK).

### Code examples

- [Actions for DynamoDB using AWS SDKs \(p. 326\)](#)
  - [Create a DynamoDB table using an AWS SDK \(p. 326\)](#)
  - [Delete a DynamoDB table using an AWS SDK \(p. 336\)](#)
  - [Delete an item from a DynamoDB table using an AWS SDK \(p. 342\)](#)
  - [Get a batch of DynamoDB items using an AWS SDK \(p. 350\)](#)
  - [Get an item from a DynamoDB table using an AWS SDK \(p. 354\)](#)
  - [Get information about a DynamoDB table \(p. 361\)](#)
  - [List DynamoDB tables using an AWS SDK \(p. 366\)](#)
  - [Put an item in a DynamoDB table using an AWS SDK \(p. 372\)](#)
  - [Query a DynamoDB table using an AWS SDK \(p. 381\)](#)
  - [Run a PartiQL statement on a DynamoDB table using an AWS SDK \(p. 393\)](#)
  - [Run batches of PartiQL statements on a DynamoDB table using an AWS SDK \(p. 405\)](#)
  - [Scan a DynamoDB table using an AWS SDK \(p. 422\)](#)
  - [Update an item in a DynamoDB table using an AWS SDK \(p. 431\)](#)
  - [Write a batch of DynamoDB items using an AWS SDK \(p. 441\)](#)
- [Scenarios for DynamoDB using AWS SDKs \(p. 449\)](#)
  - [Accelerate DynamoDB reads with DAX using an AWS SDK \(p. 449\)](#)
  - [Get started using DynamoDB tables, items, and queries using an AWS SDK \(p. 454\)](#)
  - [Query a DynamoDB table by using batches of PartiQL statements and an AWS SDK \(p. 522\)](#)
  - [Query a DynamoDB table using PartiQL and an AWS SDK \(p. 561\)](#)
- [Cross-service examples for DynamoDB using AWS SDKs \(p. 594\)](#)
  - [Build an application to submit data to a DynamoDB table \(p. 594\)](#)
  - [Create an API Gateway REST API to track COVID-19 data \(p. 595\)](#)
  - [Create a messenger application with Step Functions \(p. 596\)](#)
  - [Create a web application to track DynamoDB data \(p. 597\)](#)
  - [Create a websocket chat application with API Gateway \(p. 598\)](#)
  - [Detect PPE in images with Amazon Rekognition using an AWS SDK \(p. 599\)](#)
  - [Invoke a Lambda function from a browser \(p. 599\)](#)
  - [Save EXIF and other image information using an AWS SDK \(p. 600\)](#)
  - [Use API Gateway to invoke a Lambda function \(p. 601\)](#)

- Use Step Functions to invoke Lambda functions (p. 601)
- Use scheduled events to invoke a Lambda function (p. 602)

## Actions for DynamoDB using AWS SDKs

The following code examples show how to use Amazon DynamoDB with AWS SDKs. Each example calls an individual service function.

### Examples

- Create a DynamoDB table using an AWS SDK (p. 326)
- Delete a DynamoDB table using an AWS SDK (p. 336)
- Delete an item from a DynamoDB table using an AWS SDK (p. 342)
- Get a batch of DynamoDB items using an AWS SDK (p. 350)
- Get an item from a DynamoDB table using an AWS SDK (p. 354)
- Get information about a DynamoDB table (p. 361)
- List DynamoDB tables using an AWS SDK (p. 366)
- Put an item in a DynamoDB table using an AWS SDK (p. 372)
- Query a DynamoDB table using an AWS SDK (p. 381)
- Run a PartiQL statement on a DynamoDB table using an AWS SDK (p. 393)
- Run batches of PartiQL statements on a DynamoDB table using an AWS SDK (p. 405)
- Scan a DynamoDB table using an AWS SDK (p. 422)
- Update an item in a DynamoDB table using an AWS SDK (p. 431)
- Write a batch of DynamoDB items using an AWS SDK (p. 441)

## Create a DynamoDB table using an AWS SDK

The following code examples show how to create a DynamoDB table.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Creates a new Amazon DynamoDB table and then waits for the new
/// table to become active.
/// </summary>
/// <param name="client">An initialized Amazon DynamoDB client object.</
param>
/// <param name="tableName">The name of the table to create.</param>
/// <returns>A Boolean value indicating the success of the operation.</
returns>
    public static async Task<bool> CreateMovieTableAsync(AmazonDynamoDBClient
client, string tableName)
{
    var response = await client.CreateTableAsync(new CreateTableRequest
```

```

    {
        TableName = tableName,
        AttributeDefinitions = new List<AttributeDefinition>()
        {
            new AttributeDefinition
            {
                AttributeName = "title",
                AttributeType = "S",
            },
            new AttributeDefinition
            {
                AttributeName = "year",
                AttributeType = "N",
            },
        },
        KeySchema = new List<KeySchemaElement>()
        {
            new KeySchemaElement
            {
                AttributeName = "year",
                KeyType = "HASH",
            },
            new KeySchemaElement
            {
                AttributeName = "title",
                KeyType = "RANGE",
            },
        },
        ProvisionedThroughput = new ProvisionedThroughput
        {
            ReadCapacityUnits = 5,
            WriteCapacityUnits = 5,
        },
    });

    // Wait until the table is ACTIVE and then report success.
    Console.WriteLine("Waiting for table to become active...");

    var request = new DescribeTableRequest
    {
        TableName = response.TableDescription.TableName,
    };

    TableStatus status;

    int sleepDuration = 2000;

    do
    {
        System.Threading.Thread.Sleep(sleepDuration);

        var describeTableResponse = await
client.DescribeTableAsync(request);
        status = describeTableResponse.Table.TableStatus;

        Console.WriteLine(".");
    }
    while (status != "ACTIVE");

    return status == TableStatus.ACTIVE;
}

```

- For API details, see [CreateTable](#) in *AWS SDK for .NET API Reference*.

C++

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
if (!region.empty())
    clientConfig.region = region;
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

std::cout << "Creating table " << table <<
    " with a simple primary key: \"Name\"" << std::endl;

Aws::DynamoDB::Model::CreateTableRequest req;

Aws::DynamoDB::Model::AttributeDefinition hashKey;
hashKey.SetAttributeName("Name");
hashKey.SetAttributeType(Aws::DynamoDB::Model::ScalarAttributeType::S);
req.AddAttributeDefinitions(hashKey);

Aws::DynamoDB::Model::KeySchemaElement keyselt;

keyselt.WithAttributeName("Name").WithKeyType(Aws::DynamoDB::Model::KeyType::HASH);
req.AddKeySchema(keyselt);

Aws::DynamoDB::Model::ProvisionedThroughput thruput;
thruput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
req.SetProvisionedThroughput(thruput);
req.SetTableName(table);

const Aws::DynamoDB::Model::CreateTableOutcome& result =
dynamoClient.CreateTable(req);
if (result.IsSuccess())
{
    std::cout << "Table \""
result.GetResult().GetTableDescription().GetTableName() <<
        " created!" << std::endl;
}
else
{
    std::cout << "Failed to create table: " <<
result.GetError().GetMessage();
}
```

- For API details, see [CreateTable](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the
examples.
```

```
// It contains a DynamoDB service client that is used to act on the specified
// table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// CreateMovieTable creates a DynamoDB table with a composite primary key defined
// as
// a string sort key named `title`, and a numeric partition key named `year`.
// This function uses NewTableExistsWaiter to wait for the table to be created by
// DynamoDB before it returns.
func (basics TableBasics) CreateMovieTable() (*types.TableDescription, error) {
    var tableDesc *types.TableDescription
    table, err := basics.DynamoDbClient.CreateTable(context.TODO(),
        &dynamodb.CreateTableInput{
            AttributeDefinitions: []types.AttributeDefinition{
                {AttributeName: aws.String("year"),
                    AttributeType: types.ScalarAttributeTypeN,
                },
                {AttributeName: aws.String("title"),
                    AttributeType: types.ScalarAttributeTypeS,
                },
            },
            KeySchema: []types.KeySchemaElement{
                {AttributeName: aws.String("year"),
                    KeyType:      types.KeyTypeHash,
                },
                {AttributeName: aws.String("title"),
                    KeyType:      types.KeyTypeRange,
                },
            },
            TableName: aws.String(basics.TableName),
            ProvisionedThroughput: &types.ProvisionedThroughput{
                ReadCapacityUnits: aws.Int64(10),
                WriteCapacityUnits: aws.Int64(10),
            },
        })
    if err != nil {
        log.Printf("Couldn't create table %v. Here's why: %v\n", basics.TableName, err)
    } else {
        waiter := dynamodb.NewTableExistsWaiter(basics.DynamoDbClient)
        err = waiter.Wait(context.TODO(), &dynamodb.DescribeTableInput{
            TableName: aws.String(basics.TableName),
        }, 5*time.Minute)
        if err != nil {
            log.Printf("Wait for table exists failed. Here's why: %v\n", err)
        }
        tableDesc = table.TableDescription
    }
    return tableDesc, err
}
```

- For API details, see [CreateTable in AWS SDK for Go API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createTable(DynamoDbClient ddb, String tableName, String key) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    CreateTableRequest request = CreateTableRequest.builder()
        .attributeDefinitions(AttributeDefinition.builder()
            .attributeName(key)
            .attributeType(ScalarAttributeType.S)
            .build())
        .keySchema(KeySchemaElement.builder()
            .attributeName(key)
            .keyType(KeyType.HASH)
            .build())
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10)))
        .build())
        .tableName(tableName)
        .build();

    String newTable = "";
    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        // Wait until the Amazon DynamoDB table is created.
        WaiterResponse<DescribeTableResponse> waiterResponse =
        dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().ifPresent(System.out::println);
        newTable = response.tableDescription().tableName();
        return newTable;
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateTable in AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon DynamoDB service client object.
const ddbClient = new DynamoDBClient({ region: REGION });
export { ddbClient };
```

Create the table.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateTableCommand } from "@aws-sdk/client-dynamodb";
import { ddbClient } from "./libs/ddbClient.js";

// Set the parameters
export const params = {
  AttributeDefinitions: [
    {
      AttributeName: "Season", //ATTRIBUTE_NAME_1
      AttributeType: "N", //ATTRIBUTE_TYPE
    },
    {
      AttributeName: "Episode", //ATTRIBUTE_NAME_2
      AttributeType: "N", //ATTRIBUTE_TYPE
    },
  ],
  KeySchema: [
    {
      AttributeName: "Season", //ATTRIBUTE_NAME_1
      KeyType: "HASH",
    },
    {
      AttributeName: "Episode", //ATTRIBUTE_NAME_2
      KeyType: "RANGE",
    },
  ],
  ProvisionedThroughput: {
    ReadCapacityUnits: 1,
    WriteCapacityUnits: 1,
  },
  TableName: "TEST_TABLE", //TABLE_NAME
  StreamSpecification: {
    StreamEnabled: false,
  },
};

export const run = async () => {
  try {
    const data = await ddbClient.send(new CreateTableCommand(params));
    console.log("Table Created", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateTable](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});
```

```
// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
    AttributeDefinitions: [
        {
            AttributeName: 'CUSTOMER_ID',
            AttributeType: 'N'
        },
        {
            AttributeName: 'CUSTOMER_NAME',
            AttributeType: 'S'
        }
    ],
    KeySchema: [
        {
            AttributeName: 'CUSTOMER_ID',
            KeyType: 'HASH'
        },
        {
            AttributeName: 'CUSTOMER_NAME',
            KeyType: 'RANGE'
        }
    ],
    ProvisionedThroughput: {
        ReadCapacityUnits: 1,
        WriteCapacityUnits: 1
    },
    TableName: 'CUSTOMER_LIST',
    StreamSpecification: {
        StreamEnabled: false
    }
};

// Call DynamoDB to create the table
ddb.createTable(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Table Created", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateTable](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createNewTable(tableNameVal: String, key: String?): String? {

    val attDef = AttributeDefinition {
        attributeName = key
```

```

        attributeType = ScalarAttributeType.S
    }

    val keySchemaVal = KeySchemaElement {
        attributeName = key
        keyType = KeyType.Hash
    }

    val provisionedVal = ProvisionedThroughput {
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }

    val request = CreateTableRequest {
        attributeDefinitions = listOf(attDef)
        keySchema = listOf(keySchemaVal)
        provisionedThroughput = provisionedVal
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        var tableArn: String
        val response = ddb.createTable(request)
        ddb.waitUntilTableExists { // suspend call
            tableName = tableNameVal
        }
        tableArn = response.tableDescription!!.tableArn.toString()
        println("Table $tableArn is ready")
        return tableArn
    }
}

```

- For API details, see [CreateTable in AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a table.

```

$tableName = "ddb_demo_table_$uuid";
$service->createTable(
    $tableName,
    [
        new DynamoDBAttribute('year', 'N', 'HASH'),
        new DynamoDBAttribute('title', 'S', 'RANGE')
    ]
);

public function createTable(string $tableName, array $attributes)
{
    $keySchema = [];
    $attributeDefinitions = [];
    foreach ($attributes as $attribute) {
        if (is_a($attribute, DynamoDBAttribute::class)) {
            $keySchema[] = ['AttributeName' => $attribute->AttributeName,
'KeyType' => $attribute->KeyType];
            $attributeDefinitions[] =

```

```
[ 'AttributeName' => $attribute->AttributeName, 'AttributeType'  
=> $attribute->AttributeType];  
    }  
}  
  
$this->dynamoDbClient->createTable([  
    'TableName' => $tableName,  
    'KeySchema' => $keySchema,  
    'AttributeDefinitions' => $attributeDefinitions,  
    'ProvisionedThroughput' => ['ReadCapacityUnits' => 10,  
    'WriteCapacityUnits' => 10],  
    ]);  
}
```

- For API details, see [CreateTable in AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a table for storing movie data.

```
class Movies:  
    """Encapsulates an Amazon DynamoDB table of movie data."""  
    def __init__(self, dyn_resource):  
        """  
        :param dyn_resource: A Boto3 DynamoDB resource.  
        """  
        self.dyn_resource = dyn_resource  
        self.table = None  
  
    def create_table(self, table_name):  
        """  
        Creates an Amazon DynamoDB table that can be used to store movie data.  
        The table uses the release year of the movie as the partition key and the  
        title as the sort key.  
  
        :param table_name: The name of the table to create.  
        :return: The newly created table.  
        """  
        try:  
            self.table = self.dyn_resource.create_table(  
                TableName=table_name,  
                KeySchema=[  
                    {'AttributeName': 'year', 'KeyType': 'HASH'}, # Partition key  
                    {'AttributeName': 'title', 'KeyType': 'RANGE'} # Sort key  
                ],  
                AttributeDefinitions=[  
                    {'AttributeName': 'year', 'AttributeType': 'N'},  
                    {'AttributeName': 'title', 'AttributeType': 'S'}  
                ],  
                ProvisionedThroughput={'ReadCapacityUnits': 10,  
                'WriteCapacityUnits': 10})  
            self.table.wait_until_exists()  
        except ClientError as err:  
            logger.error(  
                "Couldn't create table %s. Here's why: %s: %s", table_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise
```

```
        else:  
            return self.table
```

- For API details, see [CreateTable in AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates an Amazon DynamoDB table that can be used to store movie data.  
# The table uses the release year of the movie as the partition key and the  
# title as the sort key.  
#  
# @param table_name [String] The name of the table to create.  
# @return [Aws::DynamoDB::Table] The newly created table.  
def create_table(table_name)  
    @table = @dynamo_resource.create_table(  
        table_name: table_name,  
        key_schema: [  
            {attribute_name: "year", key_type: "HASH"}, # Partition key  
            {attribute_name: "title", key_type: "RANGE"} # Sort key  
        ],  
        attribute_definitions: [  
            {attribute_name: "year", attribute_type: "N"},  
            {attribute_name: "title", attribute_type: "S"}  
        ],  
        provisioned_throughput: {read_capacity_units: 10, write_capacity_units: 10})  
    @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)  
rescue Aws::Errors::ServiceError => e  
    puts("Couldn't create table #{table_name}. Here's why:")  
    puts("\t#{e.code}: #{e.message}")  
    raise  
else  
    @table  
end
```

- For API details, see [CreateTable in AWS SDK for Ruby API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn create_table(client: &Client, table: &str, key: &str) -> Result<(),  
Error> {  
    let a_name: String = key.into();  
    let table_name: String = table.into();
```

```
let ad = AttributeDefinition::builder()
    .attribute_name(&a_name)
    .attribute_type(ScalarAttributeType::S)
    .build();

let ks = KeySchemaElement::builder()
    .attribute_name(&a_name)
    .key_type(KeyType::Hash)
    .build();

let pt = ProvisionedThroughput::builder()
    .read_capacity_units(10)
    .write_capacity_units(5)
    .build();

let create_table_response = client
    .create_table()
    .table_name(table_name)
    .key_schema(ks)
    .attribute_definitions(ad)
    .provisioned_throughput(pt)
    .send()
    .await;

match create_table_response {
    Ok(_) => {
        println!("Added table {} with key {}", table, key);
        Ok(())
    }
    Err(e) => {
        eprintln!("Got an error creating table:");
        eprintln!("{}: {}", "{}", e);
        Err(Error::Unhandled(Box::new(e)))
    }
}
```

- For API details, see [CreateTable in AWS SDK for Rust API reference](#).

## Delete a DynamoDB table using an AWS SDK

The following code examples show how to delete a DynamoDB table.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static async Task<bool> DeleteTableAsync(AmazonDynamoDBClient
client, string tableName)
{
    var request = new DeleteTableRequest
    {
        TableName = tableName,
    };

    var response = await client.DeleteTableAsync(request);
```

```
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Table {response.TableDescription.TableName} successfully deleted.");
            return true;
        }
        else
        {
            Console.WriteLine("Could not delete table.");
            return false;
        }
    }
```

- For API details, see [DeleteTable in AWS SDK for .NET API Reference](#).

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
if (!region.empty())
    clientConfig.region = region;
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

Aws::DynamoDB::Model::DeleteTableRequest dtr;
dtr.SetTableName(table);

const Aws::DynamoDB::Model::DeleteTableOutcome& result =
dynamoClient.DeleteTable(dtr);
if (result.IsSuccess())
{
    std::cout << "Your Table \""
result.GetResult().GetTableDescription().GetTableName() << " was deleted!\n";
}
else
{
    std::cout << "Failed to delete table: " <<
result.GetError().GetMessage();
}
```

- For API details, see [DeleteTable in AWS SDK for C++ API Reference](#).

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the examples.
```

```
// It contains a DynamoDB service client that is used to act on the specified
// table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// DeleteTable deletes the DynamoDB table and all of its data.
func (basics TableBasics) DeleteTable() error {
    _, err := basics.DynamoDbClient.DeleteTable(context.TODO(),
        &dynamodb.DeleteTableInput{
            TableName: aws.String(basics.TableName)})
    if err != nil {
        log.Printf("Couldn't delete table %v. Here's why: %v\n", basics.TableName, err)
    }
    return err
}
```

- For API details, see [DeleteTable in AWS SDK for Go API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}
```

- For API details, see [DeleteTable in AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.  
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";  
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";  
// Create an Amazon DynamoDB service client object.  
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Delete the table.

```
// Import required AWS SDK clients and commands for Node.js  
import { DeleteTableCommand } from "@aws-sdk/client-dynamodb";  
import { ddbClient } from "./libs/ddbClient.js";  
  
// Set the parameters  
export const params = {  
    TableName: "CUSTOMER_LIST_NEW",  
};  
  
export const run = async () => {  
    try {  
        const data = await ddbClient.send(new DeleteTableCommand(params));  
        console.log("Success, table deleted", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For API details, see [DeleteTable in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create the DynamoDB service object  
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});  
  
var params = {  
    TableName: process.argv[2]  
};  
  
// Call DynamoDB to delete the specified table  
ddb.deleteTable(params, function(err, data) {  
    if (err && err.code === 'ResourceNotFoundException') {  
        console.log("Error: Table not found");  
    } else if (err && err.code === 'ResourceInUseException') {  
        console.log("Error: Table in use");  
    } else {  
        console.log("Success", data);  
    }  
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteTable](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteDynamoDBTable(tableNameVal: String) {  
  
    val request = DeleteTableRequest {  
        tableName = tableNameVal  
    }  
  
    DynamoDbClient { region = "us-east-1" }.use { ddb ->  
        ddb.deleteTable(request)  
        println("$tableNameVal was deleted")  
    }  
}
```

- For API details, see [DeleteTable](#) in [AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function deleteTable(string $TableName)  
{  
    $this->customWaiter(function () use ($TableName) {  
        return $this->dynamoDbClient->deleteTable([  
            'TableName' => $TableName,  
        ]);  
    });  
}
```

- For API details, see [DeleteTable](#) in [AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None

    def delete_table(self):
        """
        Deletes the table.
        """
        try:
            self.table.delete()
            self.table = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

- For API details, see [DeleteTable in AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes the table.
def delete_table
  @table.delete
  @table = nil
rescue Aws::Errors::ServiceError => e
  puts("Couldn't delete table. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- For API details, see [DeleteTable in AWS SDK for Ruby API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_table(client: &Client, table: &str) -> Result<(), Error> {
    client.delete_table().table_name(table).send().await?;

    println!("Deleted table");

    Ok(())
}
```

- For API details, see [DeleteTable in AWS SDK for Rust API reference](#).

## Delete an item from a DynamoDB table using an AWS SDK

The following code examples show how to delete an item from a DynamoDB table.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Deletes a single item from a DynamoDB table.
///</summary>
///<param name="client">The initialized DynamoDB client object.</param>
///<param name="tableName">The name of the table from which the item
///will be deleted.</param>
///<param name="movieToDelete">A movie object containing the title and
///year of the movie to delete.</param>
///<returns>A Boolean value indicating the success or failure of the
///delete operation.</returns>
public static async Task<bool> DeleteItemAsync(
    AmazonDynamoDBClient client,
    string tableName,
    Movie movieToDelete)
{
    var key = new Dictionary<string, AttributeValue>
    {
        ["title"] = new AttributeValue { S = movieToDelete.Title },
        ["year"] = new AttributeValue { N =
movieToDelete.Year.ToString() },
    };

    var request = new DeleteItemRequest
    {
        TableName = tableName,
        Key = key,
    };

    var response = await client.DeleteItemAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [DeleteItem in AWS SDK for .NET API Reference](#).

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified
// table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// DeleteMovie removes a movie from the DynamoDB table.
func (basics TableBasics) DeleteMovie(movie Movie) error {
    _, err := basics.DynamoDbClient.DeleteItem(context.TODO(),
        &dynamodb.DeleteItemInput{
            TableName: aws.String(basics.TableName), Key: movie.GetKey(),
        })
    if err != nil {
        log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
            err)
    }
    return err
}
```

- For API details, see [DeleteItem in AWS SDK for Go API Reference](#).

Java

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName,
String key, String keyVal) {
    HashMap<String,AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    DeleteItemRequest deleteReq = DeleteItemRequest.builder()
        .tableName(tableName)
        .key(keyToGet)
        .build();

    try {
        ddb.deleteItem(deleteReq);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
```

```
        System.exit(1);
    }
}
```

- For API details, see [DeleteItem in AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: true, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {
    marshallOptions,
    unmarshallOptions,
});

export { ddbDocClient };
```

Delete an item from a table using the DynamoDB document client.

```
import { DeleteCommand } from "@aws-sdk/lib-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";
```

```
// Set the parameters.  
export const params = {  
  TableName: "TABLE_NAME",  
  Key: {  
    primaryKey: "VALUE_1",  
    sortKey: "VALUE_2",  
  },  
};  
  
export const deleteItem = async () => {  
  try {  
    await ddbDocClient.send(new DeleteCommand(params));  
    console.log("Success - item deleted");  
  } catch (err) {  
    console.log("Error", err);  
  }  
};  
deleteItem();
```

Delete an item from a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.  
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";  
import { ddbDocClient } from "../libs/ddbDocClient.js";  
  
const tableName = process.argv[2];  
const movieYear1 = process.argv[3];  
const movieTitle1 = process.argv[4];  
  
export const run = async (tableName, movieYear1, movieTitle1) => {  
  const params = {  
    Statement: "DELETE FROM " + tableName + " where title=? and year=?",  
    Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],  
  };  
  try {  
    await ddbDocClient.send(new ExecuteStatementCommand(params));  
    console.log("Success. Item deleted.");  
    return "Run successfully"; // For unit tests.  
  } catch (err) {  
    console.error(err);  
  }  
};  
run(tableName, movieYear1, movieTitle1);
```

Delete an item from a table by batch using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.  
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";  
import { ddbDocClient } from "../libs/ddbDocClient.js";  
  
const tableName = process.argv[2];  
const movieYear1 = process.argv[3];  
const movieTitle1 = process.argv[4];  
const movieYear2 = process.argv[5];  
const movieTitle2 = process.argv[6];  
  
export const run = async (  
  tableName,  
  movieYear1,
```

```
    movieTitle1,
    movieYear2,
    movieTitle2
) => {
  try {
    const params = {
      Statements: [
        {
          Statement: "DELETE FROM " + tableName + " where year=? and title=?",
          Parameters: [{ N: movieYear1 }, { S: movieTitle1 }],
        },
        {
          Statement: "DELETE FROM " + tableName + " where year=? and title=?",
          Parameters: [{ N: movieYear2 }, { S: movieTitle2 }],
        },
      ],
    };
    const data = await ddbDocClient.send(
      new BatchExecuteStatementCommand(params)
    );
    console.log("Success. Items deleted.", data);
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.error(err);
  }
};
run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteItem in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an item from a table.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
  TableName: 'TABLE',
  Key: {
    'KEY_NAME': {N: 'VALUE'}
  }
};

// Call DynamoDB to delete the item from the table
ddb.deleteItem(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

Delete an item from a table using the DynamoDB document client.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({apiVersion: '2012-08-10'});

var params = {
  Key: {
    'HASH_KEY': VALUE
  },
  TableName: 'TABLE'
};

docClient.delete(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteItem in AWS SDK for JavaScript API Reference](#).

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteDynamoDBItem(tableNameVal: String, keyName: String, keyVal: String) {

    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.S(keyVal)

    val request = DeleteItemRequest {
        tableName = tableNameVal
        key = keyToGet
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteItem(request)
        println("Item with key matching $keyVal was deleted")
    }
}
```

- For API details, see [DeleteItem in AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$key = [
    'Item' => [
        'title' => [
            'S' => $movieName,
        ],
        'year' => [
            'N' => $movieYear,
        ],
    ],
];
];

(service->deleteItemByKey($tableName, $key);
echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

public function deleteItemByKey(string $tableName, array $key)
{
    $this->dynamoDbClient->deleteItem([
        'Key' => $key['Item'],
        'TableName' => $tableName,
    ]);
}
```

- For API details, see [DeleteItem in AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None

    def delete_movie(self, title, year):
        """
        Deletes a movie from the table.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        """
        try:
            self.table.delete_item(Key={'year': year, 'title': title})
        except ClientError as err:
```

```
logger.error(
    "Couldn't delete movie %s. Here's why: %s: %s", title,
    err.response['Error']['Code'], err.response['Error']['Message'])
raise
```

You can specify a condition so that an item is deleted only when it meets certain criteria.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table

    def delete_underrated_movie(self, title, year, rating):
        """
        Deletes a movie only if it is rated below a specified value. By using a
        condition expression in a delete operation, you can specify that an item is
        deleted only when it meets certain criteria.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        :param rating: The rating threshold to check before deleting the movie.
        """
        try:
            self.table.delete_item(
                Key={'year': year, 'title': title},
                ConditionExpression="info.rating <= :val",
                ExpressionAttributeValues={":val": Decimal(str(rating))})
        except ClientError as err:
            if err.response['Error']['Code'] == "ConditionalCheckFailedException":
                logger.warning(
                    "Didn't delete %s because its rating is greater than %s.", title, rating)
            else:
                logger.error(
                    "Couldn't delete movie %s. Here's why: %s: %s", title,
                    err.response['Error']['Code'], err.response['Error']['Message'])
        raise
```

- For API details, see [DeleteItem in AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a movie from the table.
#
# @param title [String] The title of the movie to delete.
# @param year [Integer] The release year of the movie to delete.
def delete_movie(title, year)
    @table.delete_item(key: {"year" => year, "title" => title})
rescue Aws::Errors::ServiceError => e
    puts("Couldn't delete movie #{title}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
end
```

- For API details, see [DeleteItem in AWS SDK for Ruby API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_item(
    client: &Client,
    table: &str,
    key: &str,
    value: &str,
) -> Result<(), Error> {
    match client
        .delete_item()
        .table_name(table)
        .key(key, AttributeValue::S(value.into()))
        .send()
        .await
    {
        Ok(_) => {
            println!("Deleted item from table");
            Ok(())
        }
        Err(e) => Err(Error::Unhandled(Box::new(e))),
    }
}
```

- For API details, see [DeleteItem in AWS SDK for Rust API reference](#).

## Get a batch of DynamoDB items using an AWS SDK

The following code examples show how to get a batch of DynamoDB items.

## .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Gets information about an existing movie from the table.
/// </summary>
/// <param name="client">An initialized Amazon DynamoDB client object.</param>
```

```
    ///<param name="newMovie">A Movie object containing information about
    ///the movie to retrieve.</param>
    ///<param name="tableName">The name of the table containing the movie.</param>
    ///<returns>A Dictionary object containing information about the item
    ///retrieved.</returns>
    public static async Task<Dictionary<string, AttributeValue>>
GetItemAsync(AmazonDynamoDBClient client, Movie newMovie, string tableName)
{
    var key = new Dictionary<string, AttributeValue>
    {
        ["title"] = new AttributeValue { S = newMovie.Title },
        ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
    };

    var request = new GetItemRequest
    {
        Key = key,
        TableName = tableName,
    };

    var response = await client.GetItemAsync(request);
    return response.Item;
}
```

- For API details, see [BatchGetItem](#) in *AWS SDK for .NET API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon DynamoDB service client object.
const ddbClient = new DynamoDBClient({ region: REGION });
export { ddbClient };
```

Get the items.

```
// Import required AWS SDK clients and commands for Node.js
import { BatchGetItemCommand } from "@aws-sdk/client-dynamodb";
import { ddbClient } from "./libs/ddbClient.js";

// Set the parameters
export const params = {
  RequestItems: {
    TABLE_NAME: {
      Keys: [
        {
          KEY_NAME_1: { N: "KEY_VALUE" },
          KEY_NAME_2: { N: "KEY_VALUE" },
        }
      ]
    }
  }
};
```

```
        KEY_NAME_3: { N: "KEY_VALUE" },
    },
],
ProjectionExpression: "ATTRIBUTE_NAME",
},
},
};

export const run = async () => {
try {
const data = await ddbClient.send(new BatchGetItemCommand(params));
console.log("Success, items retrieved", data);
return data;
} catch (err) {
console.log("Error", err);
}
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [BatchGetItem](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
RequestItems: [
'TABLE_NAME': {
Keys: [
{'KEY_NAME': {N: 'KEY_VALUE_1'}},
{'KEY_NAME': {N: 'KEY_VALUE_2'}},
{'KEY_NAME': {N: 'KEY_VALUE_3'}}
],
ProjectionExpression: 'KEY_NAME, ATTRIBUTE'
}
]
};
};

ddb.batchGetItem(params, function(err, data) {
if (err) {
console.log("Error", err);
} else {
data.Responses.TABLE_NAME.forEach(function(element, index, array) {
console.log(element);
});
}
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [BatchGetItem](#) in [AWS SDK for JavaScript API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import decimal
import json
import logging
import os
import pprint
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
dynamodb = boto3.resource('dynamodb')

MAX_GET_SIZE = 100 # Amazon DynamoDB rejects a get batch larger than 100 items.

def do_batch_get(batch_keys):
    """
    Gets a batch of items from Amazon DynamoDB. Batches can contain keys from
    more than one table.

    When Amazon DynamoDB cannot process all items in a batch, a set of unprocessed
    keys is returned. This function uses an exponential backoff algorithm to retry
    getting the unprocessed keys until all are retrieved or the specified
    number of tries is reached.

    :param batch_keys: The set of keys to retrieve. A batch can contain at most 100
                       keys. Otherwise, Amazon DynamoDB returns an error.
    :return: The dictionary of retrieved items grouped under their respective
             table names.
    """
    tries = 0
    max_tries = 5
    sleepy_time = 1 # Start with 1 second of sleep, then exponentially increase.
    retrieved = {key: [] for key in batch_keys}
    while tries < max_tries:
        response = dynamodb.batch_get_item(RequestItems=batch_keys)
        # Collect any retrieved items and retry unprocessed keys.
        for key in response.get('Responses', []):
            retrieved[key] += response['Responses'][key]
        unprocessed = response['UnprocessedKeys']
        if len(unprocessed) > 0:
            batch_keys = unprocessed
            unprocessed_count = sum(
                [len(batch_key['Keys']) for batch_key in batch_keys.values()])
            logger.info(
                "%s unprocessed keys returned. Sleep, then retry.", unprocessed_count)
            tries += 1
            if tries < max_tries:
                logger.info("Sleeping for %s seconds.", sleepy_time)
                time.sleep(sleepy_time)
                sleepy_time = min(sleepy_time * 2, 32)
            else:
                break
    return retrieved
```

- For API details, see [BatchGetItem](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get an item from a DynamoDB table using an AWS SDK

The following code examples show how to get an item from a DynamoDB table.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);
Aws::DynamoDB::Model::GetItemRequest req;

// Set up the request.
req.SetTableName(table);
Aws::DynamoDB::Model::AttributeValue hashKey;
hashKey.SetS(keyval);
req.AddKey(key, hashKey);

// Retrieve the item's fields and values
const Aws::DynamoDB::Model::GetItemOutcome& result =
dynamoClient.GetItem(req);
if (result.IsSuccess())
{
    // Reference the retrieved fields/values.
    const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>& item
= result.GetResult().GetItem();
    if (item.size() > 0)
    {
        // Output each retrieved field and its value.
        for (const auto& i : item)
            std::cout << "Values: " << i.first << ":" << i.second.GetS()
<< std::endl;
    }
    else
    {
        std::cout << "No item found with the key " << key << std::endl;
    }
}
else
{
    std::cout << "Failed to get item: " << result.GetError().GetMessage();
}
```

- For API details, see [GetItem](#) in *AWS SDK for C++ API Reference*.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified
// table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// GetMovie gets movie data from the DynamoDB table by using the primary composite
// key
// made of title and year.
func (basics TableBasics) GetMovie(title string, year int) (Movie, error) {
    movie := Movie{Title: title, Year: year}
    response, err := basics.DynamoDbClient.GetItem(context.TODO(),
        &dynamodb.GetItemInput{
            Key: movie.GetKey(), TableName: aws.String(basics.TableName),
        })
    if err != nil {
        log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
    } else {
        err = attributevalue.UnmarshalMap(response.Item, &movie)
        if err != nil {
            log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
        }
    }
    return movie, err
}
```

- For API details, see [GetItem](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Gets an item from a table by using the DynamoDbClient.

```
public static int queryTable(DynamoDbClient ddb, String tableName, String
partitionKeyName, String partitionKeyVal, String partitionAlias) {

    // Set up an alias for the partition key name in case it's a reserved word.
    HashMap<String, String> attrNameAlias = new HashMap<String, String>();
    attrNameAlias.put(partitionAlias, partitionKeyName);

    // Set up mapping of the partition name with the value.
    HashMap<String, AttributeValue> attrValues = new HashMap<>();

    attrValues.put(": "+partitionKeyName, AttributeValue.builder()
        .s(partitionKeyVal)
        .build());

    QueryRequest queryReq = QueryRequest.builder()
        .tableName(tableName)
```

```
.keyConditionExpression(partitionAlias + " = :" + partitionKeyName)
.expressionAttributeNames(attrNameAlias)
.expressionAttributeValues(attrValues)
.build();

try {
    QueryResponse response = ddb.query(queryReq);
    return response.count();
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return -1;
}
```

- For API details, see [GetItem](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../libs/utils/util-aws-sdk.js";
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: true, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {
    marshallOptions,
    unmarshallOptions,
});
```

```
export { ddbDocClient };
```

Get an item from a table.

```
import { GetCommand } from "@aws-sdk/lib-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

// Set the parameters.
export const params = {
  TableName: "TABLE_NAME",
  Key: {
    primaryKey: "VALUE_1",
    sortKey: "VALUE_2",
  },
};

export const getItem = async () => {
  try {
    const data = await ddbDocClient.send(new GetCommand(params));
    console.log("Success :", data.Item);
  } catch (err) {
    console.log("Error", err);
  }
};
getItem();
```

Get an item from a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient";

const tableName = process.argv[2];
const movieTitle1 = process.argv[3];

export const run = async (tableName, movieTitle1) => {
  const params = {
    Statement: "SELECT * FROM " + tableName + " where title=?",
    Parameters: [{ S: movieTitle1 }],
  };
  try {
    const data = await ddbDocClient.send(new ExecuteStatementCommand(params));
    for (let i = 0; i < data.Items.length; i++) {
      console.log(
        "Success. The query return the following data. Item " + i,
        data.Items[i].year,
        data.Items[i].title,
        data.Items[i].info
      );
    }
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.error(err);
  }
};
run(tableName, movieTitle1);
```

Get items by batch from a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[5];
const movieTitle2 = process.argv[6];

export const run = async (
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2
) => {
  const params = {
    Statements: [
      {
        Statement: "SELECT * FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
      },
      {
        Statement: "SELECT * FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
      },
    ],
  };
  try {
    const data = await ddbDocClient.send(
      new BatchExecuteStatementCommand(params)
    );
    for (let i = 0; i < data.Responses.length; i++) {
      console.log(data.Responses[i].Item.year);
      console.log(data.Responses[i].Item.title);
    }
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.error(err);
  }
};
run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

- For API details, see [GetItem](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get an item from a table.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});
```

```
var params = {
  TableName: 'TABLE',
  Key: {
    'KEY_NAME': {N: '001'}
  },
  ProjectionExpression: 'ATTRIBUTE_NAME'
};

// Call DynamoDB to read the item from the table
ddb.getItem(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Item);
  }
});
```

Get an item from a table using the DynamoDB document client.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({apiVersion: '2012-08-10'});

var params = {
  TableName: 'EPISODES_TABLE',
  Key: {'KEY_NAME': VALUE}
};

docClient.get(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Item);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetItem](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getSpecificItem(tableNameVal: String, keyName: String, keyVal: String)
{
  val keyToGet = mutableMapOf<String, AttributeValue>()
```

```
keyToGet[keyName] = AttributeValue.S(keyVal)

val request = GetItemRequest {
    key = keyToGet
    tableName = tableNameVal
}

DynamoDbClient { region = "us-east-1" }.use { ddb ->
    val returnedItem = ddb.getItem(request)
    val numbersMap = returnedItem.item
    numbersMap?.forEach { key1 ->
        println(key1.key)
        println(key1.value)
    }
}
```

- For API details, see [GetItem](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$movie = $service->getItemByKey($tableName, $key);
echo "\nThe movie {$movie['Item']['title']['S']} was released in
{$movie['Item']['year']['N']}.\n";

public function getItemByKey(string $tableName, array $key)
{
    return $this->dynamoDbClient->getItem([
        'Key' => $key['Item'],
        'TableName' => $tableName,
    ]);
}
```

- For API details, see [GetItem](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None
```

```
def get_movie(self, title, year):
    """
    Gets movie data from the table for a specific movie.

    :param title: The title of the movie.
    :param year: The release year of the movie.
    :return: The data about the requested movie.
    """
    try:
        response = self.table.get_item(Key={'year': year, 'title': title})
    except ClientError as err:
        logger.error(
            "Couldn't get movie %s from table %s. Here's why: %s: %s",
            title, self.table.name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response['Item']
```

- For API details, see [GetItem](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Gets movie data from the table for a specific movie.
#
# @param title [String] The title of the movie.
# @param year [Integer] The release year of the movie.
# @return [Hash] The data about the requested movie.
def get_movie(title, year)
    response = @table.get_item(key: {"year" => year, "title" => title})
rescue Aws::Errors::ServiceError => e
    puts("Couldn't get movie #{title} from table #{@table.name}. Here's why:")
    puts("  \t#{e.code}: #{e.message}")
    raise
else
    response.item
end
```

- For API details, see [GetItem](#) in *AWS SDK for Ruby API Reference*.

## Get information about a DynamoDB table

The following code examples show how to get information about a DynamoDB table.

### C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
if (!region.empty())
    clientConfig.region = region;
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

Aws::DynamoDB::Model::DescribeTableRequest dtr;
dtr.SetTableName(table);

const Aws::DynamoDB::Model::DescribeTableOutcome& result =
dynamoClient.DescribeTable(dtr);

if (result.IsSuccess())
{
    const Aws::DynamoDB::Model::TableDescription& td =
result.GetResult().GetTable();
    std::cout << "Table name : " << td.GetTableName() << std::endl;
    std::cout << "Table ARN : " << td.GetTableArn() << std::endl;
    std::cout << "Status : " <<
Aws::DynamoDB::Model::TableStatusMapper::GetNameForTableStatus(td.GetTableStatus())
<< std::endl;
    std::cout << "Item count : " << td.GetItemCount() << std::endl;
    std::cout << "Size (bytes): " << td.GetTableSizeBytes() << std::endl;

    const Aws::DynamoDB::Model::ProvisionedThroughputDescription& ptd =
td.GetProvisionedThroughput();
    std::cout << "Throughput" << std::endl;
    std::cout << " Read Capacity : " << ptd.GetReadCapacityUnits() <<
std::endl;
    std::cout << " Write Capacity: " << ptd.GetWriteCapacityUnits() <<
std::endl;

    const Aws::Vector<Aws::DynamoDB::Model::AttributeDefinition>& ad =
td.GetAttributeDefinitions();
    std::cout << "Attributes" << std::endl;
    for (const auto& a : ad)
        std::cout << " " << a.getAttributeName() << "(" <<

Aws::DynamoDB::Model::ScalarAttributeTypeMapper::GetNameForScalarAttributeType(a.getAttributeT
<<
        ")" << std::endl;
}
else
{
    std::cout << "Failed to describe table: " <<
result.GetError().GetMessage();
}
```

- For API details, see [DescribeTable](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the
examples.
```

```
// It contains a DynamoDB service client that is used to act on the specified
// table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// TableExists determines whether a DynamoDB table exists.
func (basics TableBasics) TableExists() (bool, error) {
    exists := true
    _, err := basics.DynamoDbClient.DescribeTable(
        context.TODO(), &dynamodb.DescribeTableInput{TableName:
            aws.String(basics.TableName)},
    )
    if err != nil {
        var notFoundEx *types.ResourceNotFoundException
        if errors.As(err, &notFoundEx) {
            log.Printf("Table %v does not exist.\n", basics.TableName)
            err = nil
        } else {
            log.Printf("Couldn't determine existence of table %v. Here's why: %v\n",
                basics.TableName, err)
        }
        exists = false
    }
    return exists, err
}
```

- For API details, see [DescribeTable](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDynamoDBTable(DynamoDbClient ddb, String tableName ) {
    DescribeTableRequest request = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        TableDescription tableInfo = ddb.describeTable(request).table();
        if (tableInfo != null) {
            System.out.format("Table name : %s\n", tableInfo.tableName());
            System.out.format("Table ARN : %s\n", tableInfo.tableArn());
            System.out.format("Status : %s\n", tableInfo.tableStatus());
            System.out.format("Item count : %d\n",
                tableInfo.itemCount().longValue());
            System.out.format("Size (bytes): %d\n",
                tableInfo.tableSizeBytes().longValue());

            ProvisionedThroughputDescription throughputInfo =
                tableInfo.provisionedThroughput();
```

```
        System.out.println("Throughput");
        System.out.format("  Read Capacity : %d\n",
throughputInfo.readCapacityUnits().longValue());
        System.out.format("  Write Capacity: %d\n",
throughputInfo.writeCapacityUnits().longValue());

        List<AttributeDefinition> attributes =
tableInfo.attributeDefinitions();
        System.out.println("Attributes");

        for (AttributeDefinition a : attributes) {
            System.out.format("  %s (%s)\n", a.attributeName(),
a.attributeType());
        }
    }

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("\nDone!");
}
```

- For API details, see [DescribeTable](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon DynamoDB service client object.
const ddbClient = new DynamoDBClient({ region: REGION });
export { ddbClient };
```

Describe the table.

```
// Import required AWS SDK clients and commands for Node.js
import { DescribeTableCommand } from "@aws-sdk/client-dynamodb";
import { ddbClient } from "./libs/ddbClient.js";

// Set the parameters
export const params = { TableName: "TABLE_NAME" }; //TABLE_NAME

export const run = async () => {
    try {
        const data = await ddbClient.send(new DescribeTableCommand(params));
        console.log("Success", data);
        // console.log("Success", data.Table.KeySchema);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
}
```

```
    }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeTable](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
  TableName: process.argv[2]
};

// Call DynamoDB to retrieve the selected table descriptions
ddb.describeTable(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Table.KeySchema);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeTable](#) in [AWS SDK for JavaScript API Reference](#).

## Python

### SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None

    def exists(self, table_name):
        """
        Determines whether a table exists. As a side effect, stores the table in
```

```
a member variable.

:param table_name: The name of the table to check.
:return: True when the table exists; otherwise, False.
"""
try:
    table = self.dyn_resource.Table(table_name)
    table.load()
    exists = True
except ClientError as err:
    if err.response['Error']['Code'] == 'ResourceNotFoundException':
        exists = False
    else:
        logger.error(
            "Couldn't check for existence of %s. Here's why: %s: %s",
            table_name,
            err.response['Error']['Code'], err.response['Error']
        )
        raise
else:
    self.table = table
return exists
```

- For API details, see [DescribeTable](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Determines whether a table exists. As a side effect, stores the table in
# a member variable.
#
# @param table_name [String] The name of the table to check.
# @return [Boolean] True when the table exists; otherwise, False.
def exists?(table_name)
  table = Aws::DynamoDB::Table.new(table_name)
  table.load
  @table = table
rescue Aws::DynamoDB::Errors::ResourceNotFoundException
  puts("Table #{table_name} doesn't exist. Let's create it.")
  false
rescue Aws::Errors::ServiceError => e
  puts("Couldn't check for existence of #{table_name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  !@table.nil?
end
```

- For API details, see [DescribeTable](#) in *AWS SDK for Ruby API Reference*.

## List DynamoDB tables using an AWS SDK

The following code examples show how to list DynamoDB tables.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
listTablesRequest.SetLimit(50);
do
{
    const Aws::DynamoDB::Model::ListTablesOutcome& lto =
dynamoClient.ListTables(listTablesRequest);
    if (!lto.IsSuccess())
    {
        std::cout << "Error: " << lto.GetError().GetMessage() << std::endl;
        return 1;
    }

    for (const auto& s : lto.GetResult().GetTableNames())
        std::cout << s << std::endl;

    listTablesRequest.SetExclusiveStartTableName(lto.GetResult().GetLastEvaluatedTableName());
} while (!listTablesRequest.GetExclusiveStartTableName().empty());
```

- For API details, see [ListTables](#) in *AWS SDK for C++ API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified
// table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// ListTables lists the DynamoDB table names for the current account.
func (basics TableBasics) ListTables() ([]string, error) {
    var tableNames []string
    tables, err := basics.DynamoDbClient.ListTables(
        context.TODO(), &dynamodb.ListTablesInput{})
    if err != nil {
        log.Printf("Couldn't list tables. Here's why: %v\n", err)
    }
}
```

```
    } else {
        tableNames = tables.TableNames
    }
    return tableNames, err
}
```

- For API details, see [ListTables](#) in *AWS SDK for Go API Reference*.

Java

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllTables(DynamoDbClient ddb){

    boolean moreTables = true;
    String lastName = null;

    while(moreTables) {
        try {
            ListTablesResponse response = null;
            if (lastName == null) {
                ListTablesRequest request =
ListTablesRequest.builder().build();
                response = ddb.listTables(request);
            } else {
                ListTablesRequest request = ListTablesRequest.builder()
                    .exclusiveStartTableName(lastName).build();
                response = ddb.listTables(request);
            }

            List<String> tableNames = response.tableNames();
            if (tableNames.size() > 0) {
                for (String curName : tableNames) {
                    System.out.format("* %s\n", curName);
                }
            } else {
                System.out.println("No tables found!");
                System.exit(0);
            }

            lastName = response.lastEvaluatedTableName();
            if (lastName == null) {
                moreTables = false;
            }
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
    System.out.println("\nDone!");
}
```

- For API details, see [ListTables](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create an Amazon DynamoDB service client object.  
const ddbClient = new DynamoDBClient({ region: REGION });  
export { ddbClient };
```

List the tables.

```
// Import required AWS SDK clients and commands for Node.js  
import { ListTablesCommand } from "@aws-sdk/client-dynamodb";  
import { ddbClient } from "./libs/ddbClient.js";  
  
export const run = async () => {  
    try {  
        const data = await ddbClient.send(new ListTablesCommand({}));  
        console.log(data);  
        // console.log(data.TableNames.join("\n"));  
        return data;  
    } catch (err) {  
        console.error(err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListTables](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create the DynamoDB service object  
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});  
  
// Call DynamoDB to retrieve the list of tables  
ddb.listTables({Limit: 10}, function(err, data) {  
    if (err) {  
        console.log("Error", err.code);  
    } else {  
        console.log("Table names are ", data.TableNames);  
    }  
});
```

```
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListTables](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAllTables() {  
  
    DynamoDbClient { region = "us-east-1" }.use { ddb ->  
        val response = ddb.listTables(ListTablesRequest {})  
        response.tableNames?.forEach { tableName ->  
            println("Table name is $tableName")  
        }  
    }  
}
```

- For API details, see [ListTables](#) in [AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function listTables($exclusiveStartTableName = "", $limit = 100)  
{  
    $this->dynamoDbClient->listTables([  
        'ExclusiveStartTableName' => $exclusiveStartTableName,  
        'Limit' => $limit,  
    ]);  
}
```

- For API details, see [ListTables](#) in [AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None

    def list_tables(self):
        """
        Lists the Amazon DynamoDB tables for the current account.

        :return: The list of tables.
        """
        try:
            tables = []
            for table in self.dyn_resource.tables.all():
                print(table.name)
                tables.append(table)
        except ClientError as err:
            logger.error(
                "Couldn't list tables. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return tables
```

- For API details, see [ListTables](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_tables(client: &Client) -> Result<(), Error> {
    let paginator = client.list_tables().intoPaginator().items().send();
    let table_names = paginator.collect::<Result<Vec<_>, _>>().await?;

    println!("Tables:");

    for name in &table_names {
        println!("  {}", name);
    }

    println!("Found {} tables", table_names.len());
    Ok(())
}
```

Determine whether table exists.

```
pub async fn table_exists(client: &Client, table: &str) -> Result<bool, Error> {
```

```
    debug!("Checking for table: {table}");
    let table_list = client.list_tables().send().await;

    match table_list {
        Ok(list) =>
        Ok(list.table_names().as_ref().unwrap().contains(&table.into())),
        Err(e) => Err(Error::Unhandled(Box::new(e))),
    }
}
```

- For API details, see [ListTables](#) in *AWS SDK for Rust API reference*.

## Put an item in a DynamoDB table using an AWS SDK

The following code examples show how to put an item in a DynamoDB table.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Adds a new item to the table.
/// </summary>
/// <param name="client">An initialized Amazon DynamoDB client object.</param>
/// <param name="newMovie">A Movie object containing information for
/// the movie to add to the table.</param>
/// <param name="tableName">The name of the table where the item will be
added.</param>
/// <returns>A Boolean value that indicates the results of adding the
item.</returns>
public static async Task<bool> PutItemAsync(AmazonDynamoDBClient client,
Movie newMovie, string tableName)
{
    var item = new Dictionary<string, AttributeValue>
    {
        ["title"] = new AttributeValue { S = newMovie.Title },
        ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
    };

    var request = new PutItemRequest
    {
        TableName = tableName,
        Item = item,
    };

    var response = await client.PutItemAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [PutItem](#) in *AWS SDK for .NET API Reference*.

C++

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

Aws::DynamoDB::Model::PutItemRequest putItemRequest;
putItemRequest.SetTableName(table);

Aws::DynamoDB::Model::AttributeValue av;
av.SetS(keyVal);

Aws::DynamoDB::Model::AttributeValue album;
album.SetS(AlbumTitleValue);

Aws::DynamoDB::Model::AttributeValue awards;
awards.SetS(AwardVal);

Aws::DynamoDB::Model::AttributeValue song;
song.SetS(SongTitleVal);

// Add all AttributeValue objects.
putItemRequest.AddItem(key, av);
putItemRequest.AddItem(albumTitle, album);
putItemRequest.AddItem(Awards, awards);
putItemRequest.AddItem(SongTitle, song);

const Aws::DynamoDB::Model::PutItemOutcome result =
dynamoClient.PutItem(putItemRequest);
if (!result.IsSuccess())
{
    std::cout << result.GetError().GetMessage() << std::endl;
    return 1;
}
std::cout << "Successfully added Item!" << std::endl;
```

- For API details, see [PutItem](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified
// table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
```

```
}

// AddMovie adds a movie the DynamoDB table.
func (basics TableBasics) AddMovie(movie Movie) error {
    item, err := attributevalue.MarshalMap(movie)
    if err != nil {
        panic(err)
    }
    _, err = basics.DynamoDbClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
        TableName: aws.String(basics.TableName), Item: item,
    })
    if err != nil {
        log.Printf("Couldn't add item to table. Here's why: %v\n", err)
    }
    return err
}
```

- For API details, see [PutItem](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Puts an item into a table by using the enhanced client.

```
public static void putRecord(DynamoDbEnhancedClient enhancedClient) {
    try {
        DynamoDbTable<Customer> custTable = enhancedClient.table("Customer",
TableSchema.fromBean(Customer.class));

        // Create an Instant value.
        LocalDate localDate = LocalDate.parse("2020-04-07");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        // Populate the Table.
        Customer custRecord = new Customer();
        custRecord.setCustName("Tom red");
        custRecord.setId("id101");
        custRecord.setEmail("tred@noserver.com");
        custRecord.setRegistrationDate(instant) ;

        // Put the customer data into an Amazon DynamoDB table.
        custTable.putItem(custRecord);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Customer data added to the table with id id101");
}
```

- For API details, see [PutItem](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.  
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";  
import { DEFAULT_REGION } from "../../libs/utils/util-aws-sdk.js";  
// Create an Amazon DynamoDB service client object.  
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.  
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";  
import { ddbClient } from "./ddbClient.js";  
  
const marshallOptions = {  
    // Whether to automatically convert empty strings, blobs, and sets to `null`.  
    convertEmptyValues: false, // false, by default.  
    // Whether to remove undefined values while marshalling.  
    removeUndefinedValues: true, // false, by default.  
    // Whether to convert typeof object to map attribute.  
    convertClassInstanceToMap: false, // false, by default.  
};  
  
const unmarshallOptions = {  
    // Whether to return numbers as a string instead of converting them to native  
    // JavaScript numbers.  
    wrapNumbers: false, // false, by default.  
};  
  
// Create the DynamoDB document client.  
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {  
    marshallOptions,  
    unmarshallOptions,  
});  
  
export { ddbDocClient };
```

Put an item in a table using the DynamoDB document client.

```
import { PutCommand } from "@aws-sdk/lib-dynamodb";  
import { ddbDocClient } from "../../libs/ddbDocClient.js";  
  
export const putItem = async () => {  
    // Set the parameters.  
    export const params = {  
        TableName: "TABLE_NAME",  
        Item: {  
            primaryKey: "VALUE_1",  
            sortKey: "VALUE_2",  
        },  
    };  
    try {
```

```
    const data = await ddbDocClient.send(new PutCommand(params));
    console.log("Success - item added or updated", data);
} catch (err) {
    console.log("Error", err.stack);
}
};

putItem();
```

Put an item in a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];

export const run = async (tableName, movieTitle1, movieYear1) => {
    const params = {
        Statement: "INSERT INTO " + tableName + " value {'title':?, 'year':?}",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
    };
    try {
        await ddbDocClient.send(new ExecuteStatementCommand(params));
        console.log("Success. Item added.");
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.error(err);
    }
};
run(tableName, movieTitle1, movieYear1);
```

Put items by batch in a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[5];
const movieTitle2 = process.argv[6];

export const run = async (
    tableName,
    movieYear1,
    movieTitle1,
    movieYear2,
    movieTitle2
) => {
    const params = {
        Statements: [
            {
                Statement:
                    "INSERT INTO " + tableName + " value {'title':?, 'year':?}",
                Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
            },
            {

```

```
Statement:  
    "INSERT INTO " + tableName + " value {'title':?, 'year':?}",
    Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
},
],
];
try {
    await ddbDocClient.send(
        new BatchExecuteStatementCommand(params)
    );
    console.log("Success. Items added.");
    return "Run successfully"; // For unit tests.
} catch (err) {
    console.error(err);
}
};

run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

- For API details, see [PutItem](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Put an item in a table.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
    TableName: 'CUSTOMER_LIST',
    Item: {
        'CUSTOMER_ID' : {N: '001'},
        'CUSTOMER_NAME' : {S: 'Richard Roe'}
    }
};

// Call DynamoDB to add the item to the table
ddb.putItem(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

Put an item in a table using the DynamoDB document client.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({apiVersion: '2012-08-10'});
```

```
var params = {
  TableName: 'TABLE',
  Item: {
    'HASHKEY': VALUE,
    'ATTRIBUTE_1': 'STRING_VALUE',
    'ATTRIBUTE_2': VALUE_2
  }
};

docClient.put(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutItem](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun putItemInTable(
  tableNameVal: String,
  key: String,
  keyVal: String,
  albumTitle: String,
  albumTitleValue: String,
  awards: String,
  awardVal: String,
  songTitle: String,
  songTitleVal: String
) {
  val itemValues = mutableMapOf<String, AttributeValue>()

  // Add all content to the table.
  itemValues[key] = AttributeValue.S(keyVal)
  itemValues[songTitle] = AttributeValue.S(songTitleVal)
  itemValues[albumTitle] = AttributeValue.S(albumTitleValue)
  itemValues[awards] = AttributeValue.S(awardVal)

  val request = PutItemRequest {
    tableName = tableNameVal
    item = itemValues
  }

  DynamoDbClient { region = "us-east-1" }.use { ddb ->
    ddb.putItem(request)
    println(" A new item was placed into $tableNameVal.")
  }
}
```

```
}
```

- For API details, see [PutItem](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
echo "What's the name of the last movie you watched?\n";
while (empty($movieName)) {
    $movieName = testable_readline("Movie name: ");
}
echo "And what year was it released?\n";
$movieYear = "year";
while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
    $movieYear = testable_readline("Year released: ");
}

 getService->putItem([
    'Item' => [
        'year' => [
            'N' => "$movieYear",
        ],
        'title' => [
            'S' => $movieName,
        ],
    ],
    'TableName' => $tableName,
]);

public function putItem(array $array)
{
    $this->dynamoDbClient->putItem($array);
}
```

- For API details, see [PutItem](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None
```

```
def add_movie(self, title, year, plot, rating):
    """
    Adds a movie to the table.

    :param title: The title of the movie.
    :param year: The release year of the movie.
    :param plot: The plot summary of the movie.
    :param rating: The quality rating of the movie.
    """
    try:
        self.table.put_item(
            Item={
                'year': year,
                'title': title,
                'info': {'plot': plot, 'rating': Decimal(str(rating))}})
    except ClientError as err:
        logger.error(
            "Couldn't add movie %s to table %s. Here's why: %s: %s",
            title, self.table.name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
```

- For API details, see [PutItem](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Adds a movie to the table.
#
# @param title [String] The title of the movie.
# @param year [Integer] The release year of the movie.
# @param plot [String] The plot summary of the movie.
# @param rating [Float] The quality rating of the movie.
def add_movie(title:, year:, plot:, rating:)
    @table.put_item(
        item: {
            "year" => year,
            "title" => title,
            "info" => {"plot" => plot, "rating" => rating}})
rescue Aws::Errors::ServiceError => e
    puts("Couldn't add movie #{title} to table #{@table.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
end
```

- For API details, see [PutItem](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn add_item(client: &Client, item: Item, table: &String) -> Result<(), Error> {
    let user_av = AttributeValue::S(item.username);
    let type_av = AttributeValue::S(item.p_type);
    let age_av = AttributeValue::S(item.age);
    let first_av = AttributeValue::S(item.first);
    let last_av = AttributeValue::S(item.last);

    let request = client
        .put_item()
        .table_name(table)
        .item("username", user_av)
        .item("account_type", type_av)
        .item("age", age_av)
        .item("first_name", first_av)
        .item("last_name", last_av);

    println!("Executing request [{request:?}] to add item...");
    let resp = request.send().await?;

    let attributes = resp.attributes().unwrap();

    println!(
        "Added user {:?}, {:?} {:?}, age {:?} as {:?} user",
        attributes.get("username"),
        attributes.get("first_name"),
        attributes.get("last_name"),
        attributes.get("age"),
        attributes.get("p_type")
    );
    Ok(())
}
```

- For API details, see [PutItem](#) in *AWS SDK for Rust API reference*.

## Query a DynamoDB table using an AWS SDK

The following code examples show how to query a DynamoDB table.

.NET

### AWS SDK for .NET

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Queries the table for movies released in a particular year and
/// then displays the information for the movies returned.
/// </summary>
/// <param name="client">The initialized DynamoDB client object.</param>
```

```
/// <param name="tableName">The name of the table to query.</param>
/// <param name="year">The release year for which we want to
/// view movies.</param>
/// <returns>The number of movies that match the query.</returns>
public static async Task<int> QueryMoviesAsync(AmazonDynamoDBClient client,
string tableName, int year)
{
    var movieTable = Table.LoadTable(client, tableName);
    var filter = new QueryFilter("year", QueryOperator.Equal, year);

    Console.WriteLine("\nFind movies released in: {year}:");

    var config = new QueryOperationConfig()
    {
        Limit = 10, // 10 items per page.
        Select = SelectValues.SpecificAttributes,
        AttributesToGet = new List<string>
        {
            "title",
            "year",
        },
        ConsistentRead = true,
        Filter = filter,
    };

    // Value used to track how many movies match the
    // supplied criteria.
    var moviesFound = 0;

    Search search = movieTable.Query(config);
    do
    {
        var movieList = await search.GetNextSetAsync();
        moviesFound += movieList.Count;

        foreach (var movie in movieList)
        {
            DisplayDocument(movie);
        }
    }
    while (!search.IsDone);

    return moviesFound;
}
```

- For API details, see [Query](#) in *AWS SDK for .NET API Reference*.

C++

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;

Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);
Aws::DynamoDB::Model::QueryRequest req;

req.SetTableName(table);
```

```
// Set query key condition expression
req.SetKeyConditionExpression(partitionKeyAttributeName +
"= :valueToMatch");

// Set Expression AttributeValues
Aws::Map< Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributeValues;
attributeValues.emplace(":valueToMatch", partitionKeyValue);

req.SetExpressionAttributeValues(attributeValues);

// Perform Query operation
const Aws::DynamoDB::Model::QueryOutcome& result = dynamoClient.Query(req);
if (result.IsSuccess())
{
    // Reference the retrieved items
    const Aws::Vector< Aws::Map< Aws::String,
        Aws::DynamoDB::Model::AttributeValue>>& items = result.GetResult().GetItems();
    if(items.size() > 0)
    {
        std::cout << "Number of items retrieved from Query: " <<
        items.size() << std::endl;
        //Iterate each item and print
        for(const auto &item: items)
        {
            std::cout <<
        "*****" << std::endl;
        // Output each retrieved field and its value
        for (const auto& i : item)
            std::cout << i.first << ":" << i.second.GetS() << std::endl;
        }
    }

    else
    {
        std::cout << "No item found in table: " << table << std::endl;
    }
    else
    {
        std::cout << "Failed to Query items: " <<
        result.GetError().GetMessage();
    }
}
```

- For API details, see [Query](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified
// table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
```

```
    TableName      string
}

// Query gets all movies in the DynamoDB table that were released in the specified
// year.
// The function uses the `expression` package to build the key condition expression
// that is used in the query.
func (basics TableBasics) Query(releaseYear int) ([]Movie, error) {
    var err error
    var response *dynamodb.QueryOutput
    var movies []Movie
    keyEx := expression.Key("year").Equal(expression.Value(releaseYear))
    expr, err := expression.NewBuilder().WithKeyCondition(keyEx).Build()
    if err != nil {
        log.Printf("Couldn't build expression for query. Here's why: %v\n", err)
    } else {
        response, err = basics.DynamoDbClient.Query(context.TODO(), &dynamodb.QueryInput{
            TableName:                 aws.String(basics.TableName),
            ExpressionAttributeNames: expr.Names(),
            ExpressionAttributeValues: expr.Values(),
            KeyConditionExpression:   expr.KeyCondition(),
        })
        if err != nil {
            log.Printf("Couldn't query for movies released in %v. Here's why: %v\n",
                releaseYear, err)
        } else {
            err = attributevalue.UnmarshalListOfMaps(response.Items, &movies)
            if err != nil {
                log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
            }
        }
    }
    return movies, err
}
```

- For API details, see [Query](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Queries a table by using the enhanced client.

```
public static String queryTable(DynamoDbEnhancedClient enhancedClient) {

    try{
        DynamoDbTable<Customer> mappedTable = enhancedClient.table("Customer",
TableSchema.fromBean(Customer.class));
        QueryConditional queryConditional =
QueryConditional.keyEqualTo(Key.builder()
            .partitionValue("id101")
            .build());

        // Get items in the table and write out the ID value.
        Iterator<Customer> results =
mappedTable.query(queryConditional).items().iterator();
```

```

        String result="";
        while (results.hasNext()) {
            Customer rec = results.next();
            result = rec.getId();
            System.out.println("The record id is "+result);
        }
        return result;
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

```

Queries a table by using the enhanced client and a secondary index.

```

public static void queryIndex(DynamoDbClient ddb, String tableName) {
    try {
        // Create a DynamoDbEnhancedClient and use the DynamoDbClient object.
        DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

        //Create a DynamoDbTable object based on Movies.
        DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        String dateVal = "2013";

        DynamoDbIndex<Movies> secIndex = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class)).index("year-index");
        AttributeValue attVal = AttributeValue.builder()
        .n(dateVal)
        .build();

        // Create a QueryConditional object that's used in the query operation.
        QueryConditional queryConditional = QueryConditional
        .keyEqualTo(Key.builder().partitionValue(attVal)
        .build());

        // Get items in the table.
        SdkIterable<Page<Movies>> results =
secIndex.query(QueryEnhancedRequest.builder()
        .queryConditional(queryConditional)
        .limit(300)
        .build());

        // Display the results.
        results.forEach(page -> {
            List<Movies> allMovies = page.items();
            for (Movies myMovies: allMovies) {
                System.out.println("The movie title is " + myMovies.getTitle()
+ ". The year is " + myMovies.getYear());
            }
        });

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

Queries a table by using the DynamoDbClient.

```
public static int queryTable(DynamoDbClient ddb, String tableName, String partitionKeyName, String partitionKeyVal, String partitionAlias) {

    // Set up an alias for the partition key name in case it's a reserved word.
    HashMap<String, String> attrNameAlias = new HashMap<String, String>();
    attrNameAlias.put(partitionAlias, partitionKeyName);

    // Set up mapping of the partition name with the value.
    HashMap<String, AttributeValue> attrValues = new HashMap<>();

    attrValues.put(":" + partitionKeyName, AttributeValue.builder()
        .s(partitionKeyVal)
        .build());

    QueryRequest queryReq = QueryRequest.builder()
        .tableName(tableName)
        .keyConditionExpression(partitionAlias + " = :" + partitionKeyName)
        .expressionAttributeNames(attrNameAlias)
        .expressionAttributeValues(attrValues)
        .build();

    try {
        QueryResponse response = ddb.query(queryReq);
        return response.count();

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return -1;
}
```

Queries a table by using the DynamoDbClient and a secondary index.

```
public static void queryIndex(DynamoDbClient ddb, String tableName) {

    try {
        Map<String, String> expressionAttributesNames = new HashMap<>();
        expressionAttributesNames.put("#year", "year");
        Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(":yearValue",
        AttributeValue.builder().n("2013").build());

        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .indexName("year-index")
            .keyConditionExpression("#year = :yearValue")
            .expressionAttributeNames(expressionAttributesNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        System.out.println("== Movie Titles ==");
        QueryResponse response = ddb.query(request);
        response.items()
            .forEach(movie -> System.out.println(movie.get("title").s()));

    } catch (DynamoDbException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [Query](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon DynamoDB service client object.
const ddbClient = new DynamoDBClient({ region: REGION });
export { ddbClient };
```

Query the table.

```
// Import required AWS SDK clients and commands for Node.js
import { QueryCommand } from "@aws-sdk/client-dynamodb";
import { ddbClient } from "./libs/ddbClient.js";

// Set the parameters
export const params = {
  KeyConditionExpression: "Season = :s and Episode > :e",
  FilterExpression: "contains (Subtitle, :topic)",
  ExpressionAttributeValues: {
    ":s": { N: "1" },
    ":e": { N: "2" },
    ":topic": { S: "SubTitle" },
  },
  ProjectionExpression: "Episode, Title, Subtitle",
  TableName: "EPISODES_TABLE",
};

export const run = async () => {
  try {
    const data = await ddbClient.send(new QueryCommand(params));
    data.Items.forEach(function (element) {
      console.log(element.Title.S + (" " + element.Subtitle.S + ")");
    });
    return data;
  } catch (err) {
    console.error(err);
  }
};
run();
```

Create the client for the DynamoDB document client.

```
// Create service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: false, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

const translateConfig = { marshallOptions, unmarshallOptions };

// Create the DynamoDB Document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, translateConfig);

export { ddbDocClient };
```

Query the table using the DynamoDB document client.

```
import { QueryCommand } from "@aws-sdk/lib-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

// Set the parameters.
export const params = {
    ExpressionAttributeNames: { "#r": "rank", "#y": "year" },
    ProjectionExpression: "#r, #y, title",
    TableName: "TABLE_NAME",
    ExpressionAttributeValues: {
        ":t": "MOVIE_NAME",
        ":y": "MOVIE_YEAR",
        ":r": "MOVIE_RANK",
    },
    KeyConditionExpression: "title = :t and #y = :y",
    FilterExpression: "info.#r = :r",
};

export const queryTable = async () => {
    try {
        const data = await ddbDocClient.send(new QueryCommand(params));
        for (let i = 0; i < data.Items.length; i++) {
            console.log(
                "Success. Items with rank of " +
                "MOVIE_RANK" +
                " include\n" +
                "Year = " +
                data.Items[i].year +
                " Title = " +
                data.Items[i].title
            );
        }
    } catch (err) {
        console.log("Error", err);
    }
};
```

```
queryTable();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Query](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({apiVersion: '2012-08-10'});

var params = {
    ExpressionAttributeValues: {
        ':s': 2,
        ':e': 9,
        ':topic': 'PHRASE'
    },
    KeyConditionExpression: 'Season = :s and Episode > :e',
    FilterExpression: 'contains (Subtitle, :topic)',
    TableName: 'EPISODES_TABLE'
};

docClient.query(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data.Items);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Query](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun queryDynTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionKeyVal: String,
    partitionAlias: String
): Int {
```

```
val attrNameAlias = mutableMapOf<String, String>()
attrNameAlias[partitionAlias] = partitionKeyName

// Set up mapping of the partition name with the value.
val attrValues = mutableMapOf<String, AttributeValue>()
attrValues[":$partitionKeyName"] = AttributeValue.S(partitionKeyVal)

val request = QueryRequest {
    tableName = tableNameVal
    keyConditionExpression = "$partitionAlias = :$partitionKeyName"
    expressionAttributeNames = attrNameAlias
    this.expressionAttributeValues = attrValues
}

DynamoDbClient { region = "us-east-1" }.use { ddb ->
    val response = ddb.query(request)
    return response.count
}
}
```

- For API details, see [Query](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$birthKey = [
    'Key' => [
        'year' => [
            'N' => "$birthYear",
        ],
    ],
];
$result = $service->query($tableName, $birthKey);

public function query(string $tableName, $key)
{
    $expressionAttributeValues = [];
    $expressionAttributeNames = [];
    $keyConditionExpression = "";
    $index = 1;
    foreach ($key as $name => $value) {
        $keyConditionExpression .= "#" . array_key_first($value) . " = :"v
$index,";
        $expressionAttributeNames["#" . array_key_first($value)] =
array_key_first($value);
        $hold = array_pop($value);
        $expressionAttributeValues[":v$index"] = [
            array_key_first($hold) => array_pop($hold),
        ];
    }
    $keyConditionExpression = substr($keyConditionExpression, 0, -1);
    $query = [
        'ExpressionAttributeValues' => $expressionAttributeValues,
        'ExpressionAttributeNames' => $expressionAttributeNames,
        'KeyConditionExpression' => $keyConditionExpression,
        'TableName' => $tableName,
    ];
}
```

```
];
return $this->dynamoDbClient->query($query);
}
```

- For API details, see [Query](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Query items by using a key condition expression.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None

    def query_movies(self, year):
        """
        Queries for movies that were released in the specified year.

        :param year: The year to query.
        :return: The list of movies that were released in the specified year.
        """
        try:
            response =
self.table.query(KeyConditionExpression=Key('year').eq(year))
        except ClientError as err:
            logger.error(
                "Couldn't query for movies released in %s. Here's why: %s: %s",
year,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['Items']
```

Query items and project them to return a subset of data.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table

    def query_and_project_movies(self, year, title_bounds):
        """
        Query for movies that were released in a specified year and that have
titles
        that start within a range of letters. A projection expression is used
        to return a subset of data for each movie.

        :param year: The release year to query.
        :param title_bounds: The range of starting letters to query.
        :return: The list of movies.
        """

```

```
try:
    response = self.table.query(
        ProjectionExpression="#yr, title, info.genres, info.actors[0]",
        ExpressionAttributeNames={"#yr": "year"},
        KeyConditionExpression=
            Key('year').eq(year) &
            Key('title').between(title_bounds['first'],
        title_bounds['second'])))
    except ClientError as err:
        if err.response['Error']['Code'] == "ValidationException":
            logger.warning(
                "There's a validation error. Here's the message: %s: %s",
                err.response['Error']['Code'], err.response['Error']
            ['Message'])
        else:
            logger.error(
                "Couldn't query for movies. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']
            ['Message'])
    raise
else:
    return response['Items']
```

- For API details, see [Query](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Queries for movies that were released in the specified year.
#
# @param year [Integer] The year to query.
# @return [Array] The list of movies that were released in the specified year.
def query_movies(year)
    response = @table.query(
        key_condition_expression: "#yr = :year",
        expression_attribute_names: {"#yr" => "year"},
        expression_attribute_values: {":year" => year})
    rescue Aws::Errors::ServiceError => e
        puts("Couldn't query for movies released in #{year}. Here's why:")
        puts("\t#{e.code}: #{e.message}")
        raise
    else
        response.items
    end
```

- For API details, see [Query](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Find the movies made in the specified year.

```
pub async fn movies_in_year(
    client: &Client,
    table_name: &str,
    year: u16,
) -> Result<Vec<Movie>, MovieError> {
    let results = client
        .query()
        .table_name(table_name)
        .key_condition_expression("#yr = :yyyy")
        .expression_attribute_names("#yr", "year")
        .expression_attribute_values(":yyyy", AttributeValue::N(year.to_string()))
        .send()
        .await?;

    if let Some(items) = results.items {
        let movies = items.iter().map(|v| v.into()).collect();
        Ok(movies)
    } else {
        Ok(vec![])
    }
}
```

- For API details, see [Query](#) in *AWS SDK for Rust API reference*.

## Run a PartiQL statement on a DynamoDB table using an AWS SDK

The following code examples show how to run a PartiQL statement on a DynamoDB table.

### .NET

#### AWS SDK for .NET

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use an INSERT statement to add an item.

```
/// <summary>
/// Inserts a single movie into the movies table.
/// </summary>
/// <param name="tableName">The name of the table.</param>
/// <param name="movieTitle">The title of the movie to insert.</param>
/// <param name="year">The year that the movie was released.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the INSERT operation.</returns>
public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
{
    string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?, 'year': ?}}";
```

```
var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = insertBatch,
    Parameters = new List<AttributeValue>
    {
        new AttributeValue { S = movieTitle },
        new AttributeValue { N = year.ToString() },
    },
});
return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

Use a SELECT statement to get an item.

```
/// <summary>
/// Uses a PartiQL SELECT statement to retrieve a single movie from the
/// movie database.
/// </summary>
/// <param name="tableName">The name of the movie table.</param>
/// <param name="movieTitle">The title of the movie to retrieve.</param>
/// <returns>A list of movie data. If no movie matches the supplied
/// title, the list is empty.</returns>
public static async Task<List<Dictionary<string, AttributeValue>>>
GetSingleMovie(string tableName, string movieTitle)
{
    string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
    var parameters = new List<AttributeValue>
    {
        new AttributeValue { S = movieTitle },
    };

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = selectSingle,
    Parameters = parameters,
});
return response.Items;
}
```

Use a SELECT statement to get a list of items.

```
/// <summary>
/// Retrieve multiple movies by year using a SELECT statement.
/// </summary>
/// <param name="tableName">The name of the movie table.</param>
/// <param name="year">The year the movies were released.</param>
/// <returns></returns>
public static async Task<List<Dictionary<string, AttributeValue>>>
GetMovies(string tableName, int year)
{
    string selectSingle = $"SELECT * FROM {tableName} WHERE year = ?";
    var parameters = new List<AttributeValue>
    {
```

```

        new AttributeValue { N = year.ToString() },
    };

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = selectSingle,
    Parameters = parameters,
});

return response.Items;
}

```

Use an UPDATE statement to update an item.

```

///<summary>
/// Updates a single movie in the table, adding information for the
/// producer.
///</summary>
///<param name="tableName">the name of the table.</param>
///<param name="producer">The name of the producer.</param>
///<param name="movieTitle">The movie title.</param>
///<param name="year">The year the movie was released.</param>
///<returns>A Boolean value that indicates the success of the
/// UPDATE operation.</returns>
public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
{
    string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = insertSingle,
    Parameters = new List<AttributeValue>
    {
        new AttributeValue { S = producer },
        new AttributeValue { S = movieTitle },
        new AttributeValue { N = year.ToString() },
    },
});
    return response.StatusCode == System.Net.HttpStatusCode.OK;
}

```

Use a DELETE statement to delete a single movie.

```

///<summary>
/// Deletes a single movie from the table.
///</summary>
///<param name="tableName">The name of the table.</param>
///<param name="movieTitle">The title of the movie to delete.</param>
///<param name="year">The year that the movie was released.</param>
///<returns>A Boolean value that indicates the success of the
/// DELETE operation.</returns>
public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
{

```

```
        var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year = ?";  
  
        var response = await Client.ExecuteStatementAsync(new ExecuteStatementRequest  
        {  
            Statement = deleteSingle,  
            Parameters = new List<AttributeValue>  
            {  
                new AttributeValue { S = movieTitle },  
                new AttributeValue { N = year.ToString() },  
            },  
        });  
  
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;  
    }  
  
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use an INSERT statement to add an item.

```
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);  
  
// 2. Add a new movie using an "Insert" statement. (ExecuteStatement)  
Aws::String title;  
float rating;  
int year;  
Aws::String plot;  
{  
    title = askQuestion(  
        "Enter the title of a movie you want to add to the table: ");  
    year = askQuestionForInt("What year was it released? ");  
    rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate  
it? ",  
                                     1, 10);  
    plot = askQuestion("Summarize the plot for me: ");  
  
    Aws::DynamoDB::Model::ExecuteStatementRequest request;  
    std::stringstream sqlStream;  
    sqlStream << "INSERT INTO \" " << MOVIE_TABLE_NAME << "\" VALUE {"  
        << TITLE_KEY << ": ?, " << YEAR_KEY << ": ?, "  
        << INFO_KEY << ": ?";  
  
    request.SetStatement(sqlStream.str());  
  
    // Create the parameter attributes.  
    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;  
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));  
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));  
  
    Aws::DynamoDB::Model::AttributeValue infoMapAttribute;  
  
    std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =  
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
```

```

        ALLOCATION_TAG.c_str());
ratingAttribute->SetN(rating);
infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
    ALLOCATION_TAG.c_str());
plotAttribute->SetS(plot);
infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
attributes.push_back(infoMapAttribute);
request.SetParameters(attributes);

Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to add a movie: " <<
outcome.GetError().GetMessage()
    << std::endl;
    return false;
}
}
}

```

Use a SELECT statement to get an item.

```

// 3. Get the data for the movie using a "Select" statement.
(ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM `"
    << MOVIE_TABLE_NAME << "` WHERE "
    << TITLE_KEY << "=?" and " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to retrieve movie information: "
        << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    else {
        // Print the retrieved movie information.
        const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

        if (items.size() == 1) {
            printMovieInfo(items[0]);
        }
        else {
            std::cerr << "Error: " << items.size() << " movies were retrieved.
        }
    }
}
}

```

```
        << " There should be only one movie." << std::endl;
    }
}
```

Use an UPDATE statement to update an item.

```

// 4. Update the data for the movie using an "Update" statement.
(ExecuteStatement)
{
    rating = askQuestionForFloatRange(
        Aws::String("\nLet's update your movie.\nYou rated it ") +
        std::to_string(rating)
        + ", what new rating would you give it? ", 1, 10);

    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "UPDATE \""
        << MOVIE_TABLE_NAME << "\" SET "
        << INFO_KEY << "."
        << RATING_KEY << "=? WHERE "
        << TITLE_KEY << "=? AND "
        << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
        request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to update a movie: "
            << outcome.GetError().GetMessage();
        return false;
    }
}

```

Use a **DELETE** statement to delete an item.

```
// 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "DELETE FROM `"
        << MOVIE_TABLE_NAME << "` WHERE "
        << TITLE_KEY << "=? and "
        << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
        request);
    if (!outcome.IsSuccess()) {
```

```
        std::cerr << "Failed to delete the movie: "
        << outcome.GetError().GetMessage() << std::endl;
    return false;
}
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use an INSERT statement to add an item.

```
// AddMovie runs a PartiQL INSERT statement to add a movie to the DynamoDB table.
func (runner PartiQLRunner) AddMovie(movie Movie) error {
    params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
    movie.Info})
    if err != nil {
        panic(err)
    }
    _, err = runner.DynamoDbClient.ExecuteStatement(context.TODO(),
    &dynamodb.ExecuteStatementInput{
        Statement: aws.String(
            fmt.Sprintf("INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
            runner.TableName)),
        Parameters: params,
    })
    if err != nil {
        log.Printf("Couldn't insert an item with PartiQL. Here's why: %v\n", err)
    }
    return err
}
```

Use a SELECT statement to get an item.

```
// GetMovie runs a PartiQL SELECT statement to get a movie from the DynamoDB table
// by
// title and year.
func (runner PartiQLRunner) GetMovie(title string, year int) (Movie, error) {
    var movie Movie
    params, err := attributevalue.MarshalList([]interface{}{title, year})
    if err != nil {
        panic(err)
    }
    response, err := runner.DynamoDbClient.ExecuteStatement(context.TODO(),
    &dynamodb.ExecuteStatementInput{
        Statement: aws.String(
            fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
            runner.TableName)),
        Parameters: params,
    ))
    if err != nil {
```

```
    log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
} else {
    err = attributevalue.UnmarshalMap(response.Items[0], &movie)
    if err != nil {
        log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
    }
}
return movie, err
}
```

Use a SELECT statement to get a list of items and project the results.

```
// GetAllMovies runs a PartiQL SELECT statement to get all movies from the DynamoDB
// table.
// The results are projected to return only the title and rating of each movie.
func (runner PartiQLRunner) GetAllMovies() ([]map[string]interface{}, error) {
    var output []map[string]interface{}
    response, err := runner.DynamoDbClient.ExecuteStatement(context.TODO(),
    &dynamodb.ExecuteStatementInput{
        Statement: aws.String(
            fmt.Sprintf("SELECT title, info.rating FROM \"%v\"", runner.TableName)),
    })
    if err != nil {
        log.Printf("Couldn't get movies. Here's why: %v\n", err)
    } else {
        err = attributevalue.UnmarshalListOfMaps(response.Items, &output)
        if err != nil {
            log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
        }
    }
    return output, err
}
```

Use an UPDATE statement to update an item.

```
// UpdateMovie runs a PartiQL UPDATE statement to update the rating of a movie that
// already exists in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovie(movie Movie, rating float64) error {
    params, err := attributevalue.MarshalList([]interface{}{rating, movie.Title,
    movie.Year})
    if err != nil {
        panic(err)
    }
    _, err = runner.DynamoDbClient.ExecuteStatement(context.TODO(),
    &dynamodb.ExecuteStatementInput{
        Statement: aws.String(
            fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
            runner.TableName)),
        Parameters: params,
    })
    if err != nil {
        log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
    }
    return err
}
```

Use a `DELETE` statement to delete an item.

```
// DeleteMovie runs a PartiQL DELETE statement to remove a movie from the DynamoDB table.
func (runner PartiQLRunner) DeleteMovie(movie Movie) error {
    params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
    if err != nil {
        panic(err)
    }
    _, err = runner.DynamoDbClient.ExecuteStatement(context.TODO(),
        &dynamodb.ExecuteStatementInput{
            Statement: aws.String(
                fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
                    runner.TableName)),
            Parameters: params,
        })
    if err != nil {
        log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
            err)
    }
    return err
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Go API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
export const REGION = "eu-west-1"; // For example, "us-east-1".
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: false, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};
```

```
const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

const translateConfig = { marshallOptions, unmarshallOptions };

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, translateConfig);

export { ddbDocClient };
```

Create an item using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];

export const run = async (tableName, movieTitle1, movieYear1) => {
    const params = {
        Statement: "INSERT INTO " + tableName + " value {'title':?, 'year':?}",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
    };
    try {
        await ddbDocClient.send(new ExecuteStatementCommand(params));
        console.log("Success. Item added.");
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.error(err);
    }
}
run(tableName, movieTitle1, movieYear1);
```

Get an item using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient";

const tableName = process.argv[2];
const movieTitle1 = process.argv[3];

export const run = async (tableName, movieTitle1) => {
    const params = {
        Statement: "SELECT * FROM " + tableName + " where title=?",
        Parameters: [{ S: movieTitle1 }],
    };
    try {
        const data = await ddbDocClient.send(new ExecuteStatementCommand(params));
        for (let i = 0; i < data.Items.length; i++) {
            console.log(
                "Success. The query return the following data. Item " + i,
                data.Items[i].year,
                data.Items[i].title,
                data.Items[i].info
            );
        }
    } catch (err) {
        console.error(err);
    }
}
```

```
        );
    }
    return "Run successfully"; // For unit tests.
} catch (err) {
    console.error(err);
}
};

run(tableName, movieTitle1);
```

Update an item using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const producer1 = process.argv[5];

export const run = async (tableName, movieYear1, movieTitle1, producer1) => {
    const params = {
        Statement:
            "UPDATE " + tableName + " SET Producer=? where title=? and year=?",
        Parameters: [{ S: producer1 }, { S: movieTitle1 }, { N: movieYear1 }],
    };
    try {
        await ddbDocClient.send(new ExecuteStatementCommand(params));
        console.log("Success. Item updated.");
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.error(err);
    }
};
run(tableName, movieYear1, movieTitle1, producer1);
```

Delete an item using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];

export const run = async (tableName, movieYear1, movieTitle1) => {
    const params = {
        Statement: "DELETE FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
    };
    try {
        await ddbDocClient.send(new ExecuteStatementCommand(params));
        console.log("Success. Item deleted.");
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.error(err);
    }
};
run(tableName, movieYear1, movieTitle1);
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function insertItemByPartiQL(string $statement, array $parameters)
{
    $this->dynamoDbClient->executeStatement([
        'Statement' => "$statement",
        'Parameters' => $parameters,
    ]);
}

public function getItemByPartiQL(string $tableName, array $key): Result
{
    list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);

    return $this->dynamoDbClient->executeStatement([
        'Parameters' => $parameters,
        'Statement' => $statement,
    ]);
}

public function updateItemByPartiQL(string $statement, array $parameters)
{
    $this->dynamoDbClient->executeStatement([
        'Statement' => $statement,
        'Parameters' => $parameters,
    ]);
}

public function deleteItemByPartiQL(string $statement, array $parameters)
{
    $this->dynamoDbClient->executeStatement([
        'Statement' => $statement,
        'Parameters' => $parameters,
    ]);
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PartiQLWrapper:
    """
```

```
Encapsulates a DynamoDB resource to run PartiQL statements.  
"""\n    def __init__(self, dyn_resource):\n        """\n            :param dyn_resource: A Boto3 DynamoDB resource.\n        """\n        self.dyn_resource = dyn_resource\n\n    def run_partiql(self, statement, params):\n        """\n            Runs a PartiQL statement. A Boto3 resource is used even though\n            `execute_statement` is called on the underlying `client` object because the\n            resource transforms input and output from plain old Python objects (POPOs)\n            to\n            the DynamoDB format. If you create the client directly, you must do these\n            transforms yourself.\n\n            :param statement: The PartiQL statement.\n            :param params: The list of PartiQL parameters. These are applied to the\n                statement in the order they are listed.\n            :return: The items returned from the statement, if any.\n        """\n        try:\n            output = self.dyn_resource.meta.client.execute_statement(\n                Statement=statement, Parameters=params)\n        except ClientError as err:\n            if err.response['Error']['Code'] == 'ResourceNotFoundException':\n                logger.error(\n                    "Couldn't execute PartiQL '%s' because the table does not\nexist.",\n                    statement)\n            else:\n                logger.error(\n                    "Couldn't execute PartiQL '%s'. Here's why: %s: %s", statement,\n                    err.response['Error']['Code'], err.response['Error']\n                ['Message'])\n            raise\n        else:\n            return output
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Python (Boto3) API Reference*.

## Run batches of PartiQL statements on a DynamoDB table using an AWS SDK

The following code examples show how to run batches of PartiQL statements on a DynamoDB table.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use batches of INSERT statements to add items.

```
///<summary>\n///\n///</summary>
```

```

    ///<param name="tableName"></param>
    ///<param name="title1"></param>
    ///<param name="title2"></param>
    ///<param name="year1"></param>
    ///<param name="year2"></param>
    ///<returns></returns>
    public static async Task<bool> GetBatch(
        string tableName,
        string title1,
        string title2,
        int year1,
        int year2)
    {
        var getBatch = $"SELECT FROM {tableName} WHERE title = ? AND year = ?";
        var statements = new List<BatchStatementRequest>
        {
            new BatchStatementRequest
            {
                Statement = getBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = title1 },
                    new AttributeValue { N = year1.ToString() },
                },
            },
            new BatchStatementRequest
            {
                Statement = getBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = title2 },
                    new AttributeValue { N = year2.ToString() },
                },
            }
        };

        var response = await Client.BatchExecuteStatementAsync(new
        BatchExecuteStatementRequest
        {
            Statements = statements,
        });

        if (response.Responses.Count > 0)
        {
            response.Responses.ForEach(r =>
            {
                Console.WriteLine($"{r.Item["title"]}\t{r.Item["year"]}");
            });
            return true;
        }
        else
        {
            Console.WriteLine($"Couldn't find either {title1} or {title2}.");
            return false;
        }
    }
}

```

Use batches of SELECT statements to get items.

```
//<summary>
```

```

    ///> Summary: Inserts movies imported from a JSON file into the movie table by
    ///> using an Amazon DynamoDB PartiQL INSERT statement.
    ///>
    ///> Parameters:
    ///>   <param name="tableName">The name of the table into which the movie
    ///> information will be inserted.</param>
    ///>   <param name="movieFileName">The name of the JSON file that contains
    ///> movie information.</param>
    ///> Returns: A Boolean value that indicates the success or failure of
    ///> the insert operation.
    public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
    {
        // Get the list of movies from the JSON file.
        var movies = ImportMovies(movieFileName);

        var success = false;

        if (movies is not null)
        {
            // Insert the movies in a batch using PartiQL. Because the
            // batch can contain a maximum of 25 items, insert 25 movies
            // at a time.
            string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?, 'year': ?}}";
            var statements = new List<BatchStatementRequest>();

            try
            {
                for (var indexOffset = 0; indexOffset < 250; indexOffset += 25)
                {
                    for (var i = indexOffset; i < indexOffset + 25; i++)
                    {
                        statements.Add(new BatchStatementRequest
                        {
                            Statement = insertBatch,
                            Parameters = new List<AttributeValue>
                            {
                                new AttributeValue { S = movies[i].Title },
                                new AttributeValue { N =
movies[i].Year.ToString() },
                            },
                        });
                    }
                }

                var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
                {
                    Statements = statements,
                });

                // Wait between batches for movies to be successfully
                added.
                System.Threading.Thread.Sleep(3000);

                success = response.StatusCode ==
System.Net.HttpStatusCode.OK;

                // Clear the list of statements for the next batch.
                statements.Clear();
            }
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}

```

```

        return success;
    }

    ///<summary>
    ///<summary> Loads the contents of a JSON file into a list of movies to be
    ///</summary> added to the DynamoDB table.
    ///</summary>
    ///<param name="movieFileName">The full path to the JSON file.</param>
    ///<returns>A generic list of movie objects.</returns>
    public static List<Movie> ImportMovies(string movieFileName)
    {
        if (!File.Exists(movieFileName))
        {
            return null;
        }

        using var sr = new StreamReader(movieFileName);
        string json = sr.ReadToEnd();
        var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

        if (allMovies is not null)
        {
            // Return the first 250 entries.
            return allMovies.GetRange(0, 250);
        }
        else
        {
            return null;
        }
    }
}

```

Use batches of UPDATE statements to update items.

```

    ///<summary>
    ///<summary> Updates information for multiple movies.
    ///</summary>
    ///<param name="tableName">The name of the table containing the
    ///<param name="producer1">The producer name for the first movie
    ///<param name="title1">The title of the first movie.</param>
    ///<param name="year1">The year that the first movie was released.</param>
    ///<param name="producer2">The producer name for the second
    ///<param name="title2">The title of the second movie.</param>
    ///<param name="year2">The year that the second movie was released.</param>
    ///<returns>A Boolean value that indicates the success of the update.</returns>
    public static async Task<bool> UpdateBatch(
        string tableName,
        string producer1,
        string title1,
        int year1,
        string producer2,
        string title2,
        int year2)
    {

        string updateBatch = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";
    }
}

```

```

var statements = new List<BatchStatementRequest>
{
    new BatchStatementRequest
    {
        Statement = updateBatch,
        Parameters = new List<AttributeValue>
        {
            new AttributeValue { S = producer1 },
            new AttributeValue { S = title1 },
            new AttributeValue { N = year1.ToString() },
        },
    },
    new BatchStatementRequest
    {
        Statement = updateBatch,
        Parameters = new List<AttributeValue>
        {
            new AttributeValue { S = producer2 },
            new AttributeValue { S = title2 },
            new AttributeValue { N = year2.ToString() },
        },
    }
};

var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
{
    Statements = statements,
});

return response.StatusCode == System.Net.HttpStatusCode.OK;
}

```

Use batches of DELETE statements to delete items.

```

/// <summary>
/// Deletes multiple movies using a PartiQL BatchExecuteAsync
/// statement.
/// </summary>
/// <param name="tableName">The name of the table containing the
/// moves that will be deleted.</param>
/// <param name="title1">The title of the first movie.</param>
/// <param name="year1">The year the first movie was released.</param>
/// <param name="title2">The title of the second movie.</param>
/// <param name="year2">The year the second movie was released.</param>
/// <returns>A Boolean value indicating the success of the operation.</
returns>
public static async Task<bool> DeleteBatch(
    string tableName,
    string title1,
    int year1,
    string title2,
    int year2)
{

    string updateBatch = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";
    var statements = new List<BatchStatementRequest>
    {
        new BatchStatementRequest
        {

```

```
        Statement = updateBatch,
        Parameters = new List<AttributeValue>
        {
            new AttributeValue { S = title1 },
            new AttributeValue { N = year1.ToString() },
        },
    },

    new BatchStatementRequest
    {
        Statement = updateBatch,
        Parameters = new List<AttributeValue>
        {
            new AttributeValue { S = title2 },
            new AttributeValue { N = year2.ToString() },
        },
    },
};

var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
{
    Statements = statements,
});

return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use batches of INSERT statements to add items.

```
// 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

std::vector<Aws::String> titles;
std::vector<float> ratings;
std::vector<int> years;
std::vector<Aws::String> plots;
Aws::String doAgain = "n";
do {
    Aws::String aTitle = askQuestion(
        "Enter the title of a movie you want to add to the table: ");
    titles.push_back(aTitle);
    int aYear = askQuestionForInt("What year was it released? ");
    years.push_back(aYear);
    float aRating = askQuestionForFloatRange(
        "On a scale of 1 - 10, how do you rate it? ",
        1, 10);
    ratings.push_back(aRating);
    Aws::String aPlot = askQuestion("Summarize the plot for me: ");
    plots.push_back(aPlot);
```

```

        doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/n)
"));
    } while (doAgain == "y");

    std::cout << "Adding " << titles.size()
        << (titles.size() == 1 ? " movie " : " movies ")
        << "to the table using a batch \"INSERT\" statement." << std::endl;

{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());

    std::stringstream sqlStream;
    sqlStream << "INSERT INTO \""
        << MOVIE_TABLE_NAME << "\" VALUE {\""
        << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, "
        << INFO_KEY << "': ?}";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

        // Create attribute for the info map.
        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
            Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(ratings[i]);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
            Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
                ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plots[i]);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
        attributes.push_back(infoMapAttribute);
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;
    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
        dynamoClient.BatchExecuteStatement(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to add the movies: " <<
        outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
}

```

Use batches of SELECT statements to get items.

```
// 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM `"
        << MOVIE_TABLE_NAME << "` WHERE "
        << TITLE_KEY << "=?" and " "
        << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
    request);
    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

        for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
            const Aws::Map< Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

            printMovieInfo(item);
        }
    } else {
        std::cerr << "Failed to retrieve the movie information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}
}
```

Use batches of UPDATE statements to update items.

```
// 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

for (size_t i = 0; i < titles.size(); ++i) {
    ratings[i] = askQuestionForFloatRange(
        Aws::String("\nLet's update your the movie, `")
        + titles[i] +
        ".\nYou rated it " + std::to_string(ratings[i]) +
        ", what new rating would you give it? ", 1, 10);
}
```

```

        std::cout << "Updating the movie with a batch \"UPDATE\" statement." <<
        std::endl;

    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
            titles.size());

        std::stringstream sqlStream;
        sqlStream << "UPDATE \" " << MOVIE_TABLE_NAME << "\" SET "
        << INFO_KEY << "." << RATING_KEY << "=? WHERE "
        << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);

            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
            attributes.push_back(
                Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

            attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);
        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
        dynamoClient.BatchExecuteStatement(
            request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update movie information: "
            << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }
}

```

Use batches of DELETE statements to delete items.

```

// 6. Delete multiple movies using "Delete" statements. (BatchExecuteStatement)
{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "DELETE FROM \" " << MOVIE_TABLE_NAME << "\" WHERE "
    << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;
}

```

```
    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete the movies: "
        << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for C++ API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use batches of INSERT statements to add items.

```
// AddMovieBatch runs a batch of PartiQL INSERT statements to add multiple movies
// to the
// DynamoDB table.
func (runner PartiQLRunner) AddMovieBatch(movies []Movie) error {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
        movie.Info})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(fmt.Sprintf(
                "INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
                runner.TableName)),
            Parameters: params,
        }
    }

    _, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
    &dynamodb.BatchExecuteStatementInput{
        Statements: statementRequests,
    })
    if err != nil {
        log.Printf("Couldn't insert a batch of items with PartiQL. Here's why: %v\n",
        err)
    }
    return err
}
```

Use batches of SELECT statements to get items.

```
// GetMovieBatch runs a batch of PartiQL SELECT statements to get multiple movies
// from
// the DynamoDB table by title and year.
func (runner PartiQLRunner) GetMovieBatch(movies []Movie) ([]Movie, error) {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(
                fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
                    runner.TableName)),
            Parameters: params,
        }
    }

    output, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
        &dynamodb.BatchExecuteStatementInput{
            Statements: statementRequests,
        })
    var outMovies []Movie
    if err != nil {
        log.Printf("Couldn't get a batch of items with PartiQL. Here's why: %v\n", err)
    } else {
        for _, response := range output.Responses {
            var movie Movie
            err = attributevalue.UnmarshalMap(response.Item, &movie)
            if err != nil {
                log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
            } else {
                outMovies = append(outMovies, movie)
            }
        }
    }
    return outMovies, err
}
```

Use batches of UPDATE statements to update items.

```
// UpdateMovieBatch runs a batch of PartiQL UPDATE statements to update the rating
// of
// multiple movies that already exist in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovieBatch(movies []Movie, ratings []float64) error {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{ratings[index],
            movie.Title, movie.Year})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(
                fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
                    runner.TableName)),
            Parameters: params,
        }
    }
```

```
_ , err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
&dynamodb.BatchExecuteStatementInput{
    Statements: statementRequests,
})
if err != nil {
    log.Printf("Couldn't update the batch of movies. Here's why: %v\n", err)
}
return err
}
```

Use batches of DELETE statements to delete items.

```
// DeleteMovieBatch runs a batch of PartiQL DELETE statements to remove multiple
// movies from the DynamoDB table.
func (runner PartiQLRunner) DeleteMovieBatch(movies []Movie) error {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(
                fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
            Parameters: params,
        }
    }

    _, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
&dynamodb.BatchExecuteStatementInput{
    Statements: statementRequests,
})
    if err != nil {
        log.Printf("Couldn't delete the batch of movies. Here's why: %v\n", err)
    }
    return err
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Go API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
export const REGION = "eu-west-1"; // For example, "us-east-1".
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: false, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

const translateConfig = { marshallOptions, unmarshallOptions };

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, translateConfig);

export { ddbDocClient };
```

Create a batch of items using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[5];
const movieTitle2 = process.argv[6];

export const run = async (
    tableName,
    movieYear1,
    movieTitle1,
    movieYear2,
    movieTitle2
) => {
    const params = {
        Statements: [
            {
                Statement:
                    "INSERT INTO " + tableName + " value {'title':?, 'year':?}",
                Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
            },
            {
                Statement:
                    "INSERT INTO " + tableName + " value {'title':?, 'year':?}",
                Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
            },
        ],
    };
}
```

```
        ],
    };
    try {
        await ddbDocClient.send(
            new BatchExecuteStatementCommand(params)
        );
        console.log("Success. Items added.");
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.error(err);
    }
};

run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

Get a batch of items using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[5];
const movieTitle2 = process.argv[6];

export const run = async (
    tableName,
    movieYear1,
    movieTitle1,
    movieYear2,
    movieTitle2
) => {
    const params = {
        Statements: [
            {
                Statement: "SELECT * FROM " + tableName + " where title=? and year=?",
                Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
            },
            {
                Statement: "SELECT * FROM " + tableName + " where title=? and year=?",
                Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
            },
        ],
    };
    try {
        const data = await ddbDocClient.send(
            new BatchExecuteStatementCommand(params)
        );
        for (let i = 0; i < data.Responses.length; i++) {
            console.log(data.Responses[i].Item.year);
            console.log(data.Responses[i].Item.title);
        }
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.error(err);
    }
};

run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

Update a batch of items using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[6];
const movieTitle2 = process.argv[7];
const producer1 = process.argv[5];
const producer2 = process.argv[8];

export const run = async (
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2,
  producer1,
  producer2
) => {
  const params = {
    Statements: [
      {
        Statement:
          "UPDATE " + tableName + " SET producer=? where title=? and year=?",
        Parameters: [{ S: producer1 }, { S: movieTitle1 }, { N: movieYear1 }],
      },
      {
        Statement:
          "UPDATE " + tableName + " SET producer=? where title=? and year=?",
        Parameters: [{ S: producer2 }, { S: movieTitle2 }, { N: movieYear2 }],
      },
    ],
  };
  try {
    await ddbDocClient.send(
      new BatchExecuteStatementCommand(params)
    );
    console.log("Success. Items updated.");
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.error(err);
  }
};
run(
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2,
  producer1,
  producer2
);
```

Delete a batch of items using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
```

```
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[5];
const movieTitle2 = process.argv[6];

export const run = async (
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2
) => {
  try {
    const params = {
      Statements: [
        {
          Statement: "DELETE FROM " + tableName + " where year=? and title=?",
          Parameters: [{ N: movieYear1 }, { S: movieTitle1 }],
        },
        {
          Statement: "DELETE FROM " + tableName + " where year=? and title=?",
          Parameters: [{ N: movieYear2 }, { S: movieTitle2 }],
        },
      ],
    };
    const data = await ddbDocClient.send(
      new BatchExecuteStatementCommand(params)
    );
    console.log("Success. Items deleted.", data);
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.error(err);
  }
};
run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function getItemByPartiQLBatch(string $tableName, array $keys): Result
{
    $statements = [];
    foreach ($keys as $key) {
        list($statement, $parameters) = $this->buildStatementAndParameters("SELECT", $tableName, $key['Item']);
        $statements[] = [
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ];
    }

    return $this->dynamoDbClient->batchExecuteStatement([
        'Statements' => $statements,
    ]);
}
```

```

        }

    public function insertItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
}

```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class PartiQLBatchWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource

    def run_partiql(self, statements, param_list):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because the

```

```

    to
        resource transforms input and output from plain old Python objects (POPOs)
        to
            the DynamoDB format. If you create the client directly, you must do these
            transforms yourself.

            :param statements: The batch of PartiQL statements.
            :param param_list: The batch of PartiQL parameters that are associated with
                each statement. This list must be in the same order as
            the
                statements.

            :return: The responses returned from running the statements, if any.
            """
        try:
            output = self.dyn_resource.meta.client.batch_execute_statement(
                Statements=[{
                    'Statement': statement, 'Parameters': params
                } for statement, params in zip(statements, param_list)])
        except ClientError as err:
            if err.response['Error']['Code'] == 'ResourceNotFoundException':
                logger.error(
                    "Couldn't execute batch of PartiQL statements because the table
"
                    "does not exist.")
            else:
                logger.error(
                    "Couldn't execute batch of PartiQL statements. Here's why: %s:
%s",
                    err.response['Error']['Code'], err.response['Error']
                ['Message'])
                raise
        else:
            return output

```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scan a DynamoDB table using an AWS SDK

The following code examples show how to scan a DynamoDB table.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

public static async Task<int> ScanTableAsync(
    AmazonDynamoDBClient client,
    string tableName,
    int startYear,
    int endYear)
{
    var request = new ScanRequest
    {
        TableName = tableName,
        ExpressionAttributeNames = new Dictionary<string, string>
        {
            { "#yr", "year" },
        },
        ExpressionAttributeValues = new Dictionary<string, AttributeValue>

```

```
{  
    { ":y_a", new AttributeValue { N = startYear.ToString() } },  
    { ":y_z", new AttributeValue { N = endYear.ToString() } },  
},  
FilterExpression = "#yr between :y_a and :y_z",  
ProjectionExpression = "#yr, title, info.actors[0], info.directors,  
info.running_time_secs",  
};  
  
// Keep track of how many movies were found.  
int foundCount = 0;  
  
var response = new ScanResponse();  
do  
{  
    response = await client.ScanAsync(request);  
    foundCount += response.Items.Count;  
    response.Items.ForEach(i => DisplayItem(i));  
}  
while (response.LastEvaluatedKey.Count > 1);  
return foundCount;  
}
```

- For API details, see [Scan](#) in [AWS SDK for .NET API Reference](#).

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;  
  
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);  
Aws::DynamoDB::Model::ScanRequest req;  
req.SetTableName(table);  
  
if (!projection.empty())  
    req.SetProjectionExpression(projection);  
  
// Perform scan on table  
const Aws::DynamoDB::Model::ScanOutcome& result = dynamoClient.Scan(req);  
if (result.IsSuccess())  
{  
    // Reference the retrieved items  
    const Aws::Vector<Aws::Map<Aws::String,  
Aws::DynamoDB::Model::AttributeValue>>& items = result.GetResult().GetItems();  
    if(items.size() > 0)  
    {  
        std::cout << "Number of items retrieved from scan: " <<  
items.size() << std::endl;  
        //Iterate each item and print  
        for(const auto &item: items)  
        {  
            std::cout <<  
"*****" << std::endl;  
            // Output each retrieved field and its value  
            for (const auto& i : item)  
                std::cout << i.first << ":" << i.second.GetS() << std::endl;
```

```
        }
    }

    else
    {
        std::cout << "No item found in table: " << table << std::endl;
    }
else
{
    std::cout << "Failed to Scan items: " <<
result.GetError().GetMessage();
}
```

- For API details, see [Scan](#) in *AWS SDK for C++ API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified
// table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// Scan gets all movies in the DynamoDB table that were released in a range of
// years
// and projects them to return a reduced set of fields.
// The function uses the `expression` package to build the filter and projection
// expressions.
func (basics TableBasics) Scan(startYear int, endYear int) ([]Movie, error) {
    var movies []Movie
    var err error
    var response *dynamodb.ScanOutput
    filtEx := expression.Name("year").Between(expression.Value(startYear),
        expression.Value(endYear))
    projEx := expression.NamesList(
        expression.Name("year"), expression.Name("title"),
        expression.Name("info.rating"))
    expr, err :=
        expression.NewBuilder().WithFilter(filtEx).WithProjection(projEx).Build()
    if err != nil {
        log.Printf("Couldn't build expressions for scan. Here's why: %v\n", err)
    } else {
        response, err = basics.DynamoDbClient.Scan(context.TODO(), &dynamodb.ScanInput{
            TableName:                 aws.String(basics.TableName),
            ExpressionAttributeNames: expr.Names(),
            ExpressionAttributeValues: expr.Values(),
            FilterExpression:         expr.Filter(),
            ProjectionExpression:     expr.Projection(),
        })
    }
}
```

```
    if err != nil {
        log.Printf("Couldn't scan for movies released between %v and %v. Here's why: %v
\n",
        startYear, endYear, err)
    } else {
        err = attributevalue.UnmarshalListOfMaps(response.Items, &movies)
        if err != nil {
            log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
        }
    }
}
return movies, err
}
```

- For API details, see [Scan](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Scans an Amazon DynamoDB table by using the enhanced client.

```
public static void scan( DynamoDbEnhancedClient enhancedClient) {
    try{
        DynamoDbTable<Customer> custTable = enhancedClient.table("Customer",
TableSchema.fromBean(Customer.class));
        Iterator<Customer> results = custTable.scan().items().iterator();
        while (results.hasNext()) {
            Customer rec = results.next();
            System.out.println("The record id is "+rec.getId());
            System.out.println("The name is " +rec.getCustName());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

- For API details, see [Scan](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../libs/utils/util-aws-sdk.js";
```

```
// Create an Amazon DynamoDB service client object.  
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.  
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";  
import { ddbClient } from "./ddbClient.js";  
  
const marshallOptions = {  
    // Whether to automatically convert empty strings, blobs, and sets to `null`.  
    convertEmptyValues: false, // false, by default.  
    // Whether to remove undefined values while marshalling.  
    removeUndefinedValues: true, // false, by default.  
    // Whether to convert typeof object to map attribute.  
    convertClassInstanceToMap: false, // false, by default.  
};  
  
const unmarshallOptions = {  
    // Whether to return numbers as a string instead of converting them to native  
    // JavaScript numbers.  
    wrapNumbers: false, // false, by default.  
};  
  
// Create the DynamoDB document client.  
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {  
    marshallOptions,  
    unmarshallOptions,  
});  
  
export { ddbDocClient };
```

Scan a table using the DynamoDB document client.

```
// Import required AWS SDK clients and commands for Node.js.  
import { ddbDocClient } from "../libs/ddbDocClient.js";  
import { ScanCommand } from "@aws-sdk/lib-dynamodb";  
// Set the parameters.  
export const params = {  
    TableName: "TABLE_NAME",  
    ProjectionExpression: "#r, #y, title",  
    ExpressionAttributeNames: { "#r": "rank", "#y": "year" },  
    FilterExpression: "title = :t and #y = :y and info.#r = :r",  
    ExpressionAttributeValues: {  
        ":r": "MOVIE_RANK",  
        ":y": "MOVIE_YEAR",  
        ":t": "MOVIE_NAME",  
    },  
};  
  
export const scanTable = async () => {  
    try {  
        const data = await ddbDocClient.send(new ScanCommand(params));  
        console.log("success", data.Items);  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
scanTable();
```

- For API details, see [Scan in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js.
var AWS = require("aws-sdk");
// Set the AWS Region.
AWS.config.update({ region: "REGION" });

// Create DynamoDB service object.
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

const params = {
    // Specify which items in the results are returned.
    FilterExpression: "Subtitle = :topic AND Season = :s AND Episode = :e",
    // Define the expression attribute value, which are substitutes for the values
    // you want to compare.
    ExpressionAttributeValues: {
        ":topic": {S: "SubTitle2"},
        ":s": {N: 1},
        ":e": {N: 2},
    },
    // Set the projection expression, which are the attributes that you want.
    ProjectionExpression: "Season, Episode, Title, Subtitle",
    TableName: "EPISODES_TABLE",
};

ddb.scan(params, function (err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
        data.Items.forEach(function (element, index, array) {
            console.log(
                "printing",
                element.Title.S + " (" + element.Subtitle.S + ")"
            );
        });
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Scan in AWS SDK for JavaScript API Reference](#).

## Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun scanItems(tableNameVal: String) {

    val request = ScanRequest {
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}
```

- For API details, see [Scan](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$yearsKey = [
    'Key' => [
        'year' => [
            'N' => [
                'minRange' => 1990,
                'maxRange' => 1999,
            ],
        ],
    ],
];
$filter = "year between 1990 and 1999";
echo "\nHere's a list of all the movies released in the 90s:\n";
$result = $service->scan($tableName, $yearsKey, $filter);
foreach ($result['Items'] as $movie) {
    $movie = $marshal->unmarshalItem($movie);
    echo $movie['title'] . "\n";
}

public function scan(string $tableName, array $key, string $filters)
{
    $query = [
        'ExpressionAttributeNames' => ['#year' => 'year'],
        'ExpressionAttributeValues' => [
            ":min" => ['N' => '1990'],
            ":max" => ['N' => '1999'],
        ],
        'FilterExpression' => "#year between :min and :max",
        'TableName' => $tableName,
    ];
    return $this->dynamoDbClient->scan($query);
}
```

- For API details, see [Scan](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:  
    """Encapsulates an Amazon DynamoDB table of movie data."""  
    def __init__(self, dyn_resource):  
        """  
        :param dyn_resource: A Boto3 DynamoDB resource.  
        """  
        self.dyn_resource = dyn_resource  
        self.table = None  
  
    def scan_movies(self, year_range):  
        """  
        Scans for movies that were released in a range of years.  
        Uses a projection expression to return a subset of data for each movie.  
  
        :param year_range: The range of years to retrieve.  
        :return: The list of movies released in the specified years.  
        """  
        movies = []  
        scan_kwargs = {  
            'FilterExpression': Key('year').between(year_range['first'],  
year_range['second']),  
            'ProjectionExpression': "#yr, title, info.rating",  
            'ExpressionAttributeNames': {"#yr": "year"}  
        }  
        try:  
            done = False  
            start_key = None  
            while not done:  
                if start_key:  
                    scan_kwargs['ExclusiveStartKey'] = start_key  
                response = self.table.scan(**scan_kwargs)  
                movies.extend(response.get('Items', []))  
                start_key = response.get('LastEvaluatedKey', None)  
                done = start_key is None  
            except ClientError as err:  
                logger.error(  
                    "Couldn't scan for movies. Here's why: %s: %s",  
                    err.response['Error']['Code'], err.response['Error']['Message'])  
                raise  
  
        return movies
```

- For API details, see [Scan](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Scans for movies that were released in a range of years.
# Uses a projection expression to return a subset of data for each movie.
#
# @param year_range [Hash] The range of years to retrieve.
# @return [Array] The list of movies released in the specified years.
def scan_movies(year_range)
  movies = []
  scan_hash = {
    filter_expression: "#yr between :start_yr and :end_yr",
    projection_expression: "#yr, title, info.rating",
    expression_attribute_names: {"#yr" => "year"},
    expression_attribute_values: {
      ":start_yr" => year_range[:start], ":end_yr" => year_range[:end]}
  }
  done = false
  start_key = nil
  until done
    scan_hash[:exclusive_start_key] = start_key unless start_key.nil?
    response = @table.scan(scan_hash)
    movies.concat(response.items) unless response.items.nil?
    start_key = response.last_evaluated_key
    done = start_key.nil?
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't scan for movies. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  movies
end
```

- For API details, see [Scan](#) in [AWS SDK for Ruby API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_items(client: &Client, table: &str) -> Result<(), Error> {
    let items: Result<Vec<_>, _> = client
        .scan()
        .table_name(table)
        .into_paginator()
        .items()
        .send()
        .collect()
        .await;

    println!("Items in table:");
    for item in items? {
        println!("  {:?}", item);
    }
}

Ok()
```

```
}
```

- For API details, see [Scan](#) in *AWS SDK for Rust API reference*.

## Update an item in a DynamoDB table using an AWS SDK

The following code examples show how to update an item in a DynamoDB table.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Updates an existing item in the movies table.
///</summary>
///<param name="client">An initialized Amazon DynamoDB client object.</param>
///<param name="newMovie">A Movie object containing information for the movie to update.</param>
///<param name="newInfo">A MovieInfo object that contains the information that will be changed.</param>
///<param name="tableName">The name of the table that contains the movie.</param>
///<returns>A Boolean value that indicates the success of the operation.</returns>
public static async Task<bool> UpdateItemAsync(
    AmazonDynamoDBClient client,
    Movie newMovie,
    MovieInfo newInfo,
    string tableName)
{
    var key = new Dictionary<string, AttributeValue>
    {
        ["title"] = new AttributeValue { S = newMovie.Title },
        ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
    };
    var updates = new Dictionary<string, AttributeValueUpdate>
    {
        ["info.plot"] = new AttributeValueUpdate
        {
            Action = AttributeAction.PUT,
            Value = new AttributeValue { S = newInfo.Plot },
        },
        ["info.rating"] = new AttributeValueUpdate
        {
            Action = AttributeAction.PUT,
            Value = new AttributeValue { N = newInfo.Rank.ToString() },
        },
    };

    var request = new UpdateItemRequest
    {
        AttributeUpdates = updates,
        Key = key,
        TableName = tableName,
```

```
};

var response = await client.UpdateItemAsync(request);

return response.HttpStatusCode == System.Net HttpStatusCode.OK;
}
```

- For API details, see [UpdateItem](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

// *** Define UpdateItem request arguments
// Define TableName argument.
Aws::DynamoDB::Model::UpdateItemRequest request;
request.SetTableName(tableName);

// Define KeyName argument.
Aws::DynamoDB::Model::AttributeValue attribValue;
attribValue.SetS(keyValue);
request.AddKey("id", attribValue);

// Construct the SET update expression argument.
Aws::String update_expression("SET #a = :valueA");
request.SetUpdateExpression(update_expression);

// Parse the attribute name and value. Syntax: "name=value".
auto parsed = Aws::Utils::StringUtils::Split(attributeNameAndValue, '=');

if (parsed.size() != 2)
{
    std::cout << "Invalid argument syntax: " << attributeNameAndValue <<
USAGE;
    return 1;
}

// Construct attribute name argument
// Note: Setting the ExpressionAttributeNames argument is required only
// when the name is a reserved word, such as "default". Otherwise, the
// name can be included in the update_expression, as in
// "SET MyAttributeName = :valueA"
Aws::Map<Aws::String, Aws::String> expressionAttributeNames;
expressionAttributeNames["#a"] = parsed[0];
request.SetExpressionAttributeNames(expressionAttributeNames);

// Construct attribute value argument.
Aws::DynamoDB::Model::AttributeValue attributeUpdatedValue;
attributeUpdatedValue.SetS(parsed[1]);
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
expressionAttributeValues;
expressionAttributeValues[":valueA"] = attributeUpdatedValue;
request.SetExpressionAttributeValues(expressionAttributeValues);
```

```
// Update the item.
const Aws::DynamoDB::Model::UpdateItemOutcome& result =
dynamoClient.UpdateItem(request);
if (!result.IsSuccess())
{
    std::cout << result.GetError().GetMessage() << std::endl;
    return 1;
}
std::cout << "Item was updated" << std::endl;
```

- For API details, see [UpdateItem in AWS SDK for C++ API Reference](#).

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified
// table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// UpdateMovie updates the rating and plot of a movie that already exists in the
// DynamoDB table. This function uses the `expression` package to build the update
// expression.
func (basics TableBasics) UpdateMovie(movie Movie)
(map[string]map[string]interface{}, error) {
var err error
var response *dynamodb.UpdateItemOutput
var attributeMap map[string]map[string]interface{}
update := expression.Set(expression.Name("info.rating"),
expression.Value(movie.Info["rating"]))
update.Set(expression.Name("info.plot"), expression.Value(movie.Info["plot"]))
expr, err := expression.NewBuilder().WithUpdate(update).Build()
if err != nil {
    log.Printf("Couldn't build expression for update. Here's why: %v\n", err)
} else {
    response, err = basics.DynamoDbClient.UpdateItem(context.TODO(),
&dynamodb.UpdateItemInput{
    TableName:           aws.String(basics.TableName),
    Key:                movie.GetKey(),
    ExpressionAttributeNames: expr.Names(),
    ExpressionAttributeValues: expr.Values(),
    UpdateExpression:      expr.Update(),
    ReturnValue:          types.ReturnValueUpdatedNew,
})
if err != nil {
    log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
} else {
    err = attributevalue.UnmarshalMap(response.Attributes, &attributeMap)
    if err != nil {
        log.Printf("Couldn't unmarshall update response. Here's why: %v\n", err)
    }
}
}
```

```
        }
    }
    return attributeMap, err
}
```

- For API details, see [UpdateItem](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Updates an item located in a table by using the enhanced client.

```
public static String modifyItem(DynamoDbEnhancedClient enhancedClient, String keyVal, String email) {

    try {

        DynamoDbTable<Customer> mappedTable = enhancedClient.table("Customer",
TableSchema.fromBean(Customer.class));
        Key key = Key.builder()
            .partitionValue(keyVal)
            .build();

        // Get the item by using the key and update the email value.
        Customer customerRec = mappedTable.getItem(r->r.key(key));
        customerRec.setEmail(email);
        mappedTable.updateItem(customerRec);
        return customerRec.getEmail();

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [UpdateItem](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../libs/utils/util-aws-sdk.js";
// Create an Amazon DynamoDB service client object.
```

```
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: true, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {
    marshallOptions,
    unmarshallOptions,
});

export { ddbDocClient };
```

Update an item in a table using the DynamoDB document client.

```
import { UpdateCommand } from "@aws-sdk/lib-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

export const updateItem = async () => {
    // Set the parameters.
    const params = {
        TableName: "TABLE_NAME",
        Key: {
            title: "MOVIE_NAME",
            year: "MOVIE_YEAR",
        },
        ProjectionExpression: "#r",
        ExpressionAttributeNames: { "#r": "rank" },
        UpdateExpression: "set info.plot = :p, info.#r = :r",
        ExpressionAttributeValues: {
            ":p": "MOVIE_PLOT",
            ":r": "MOVIE_RANK",
        },
    };
    try {
        const data = await ddbDocClient.send(new UpdateCommand(params));
        console.log("Success - item added or updated", data);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
```

```
updateItem();
```

Update an item in a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const producer1 = process.argv[5];

export const run = async (tableName, movieYear1, movieTitle1, producer1) => {
  const params = {
    Statement:
      "UPDATE " + tableName + " SET Producer=? where title=? and year=?",
    Parameters: [{ S: producer1 }, { S: movieTitle1 }, { N: movieYear1 }],
  };
  try {
    await ddbDocClient.send(new ExecuteStatementCommand(params));
    console.log("Success. Item updated.");
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.error(err);
  }
};
run(tableName, movieYear1, movieTitle1, producer1);
```

Update items by batch in a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[6];
const movieTitle2 = process.argv[7];
const producer1 = process.argv[5];
const producer2 = process.argv[8];

export const run = async (
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2,
  producer1,
  producer2
) => {
  const params = {
    Statements: [
      {
        Statement:
          "UPDATE " + tableName + " SET producer=? where title=? and year=?",
        Parameters: [{ S: producer1 }, { S: movieTitle1 }, { N: movieYear1 }],
      },
    ],
  };
}
```

```
{  
    Statement:  
        "UPDATE " + tableName + " SET producer=? where title=? and year=?",  
        Parameters: [{ S: producer2 }, { S: movieTitle2 }, { N: movieYear2 }],  
    },  
},  
];  
try {  
    await ddbDocClient.send(  
        new BatchExecuteStatementCommand(params)  
    );  
    console.log("Success. Items updated.");  
    return "Run successfully"; // For unit tests.  
} catch (err) {  
    console.error(err);  
}  
};  
run(  
    tableName,  
    movieYear1,  
    movieTitle1,  
    movieYear2,  
    movieTitle2,  
    producer1,  
    producer2  
);
```

- For API details, see [UpdateItem in AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun updateTableItem(  
    tableNameVal: String,  
    keyName: String,  
    keyVal: String,  
    name: String,  
    updateVal: String  
) {  
  
    val itemKey = mutableMapOf<String, AttributeValue>()  
    itemKey[keyName] = AttributeValue.S(keyVal)  
  
    val updatedValues = mutableMapOf<String, AttributeValueUpdate>()  
    updatedValues[name] = AttributeValueUpdate {  
        value = AttributeValue.S(updateVal)  
        action = AttributeAction.Put  
    }  
  
    val request = UpdateItemRequest {  
        tableName = tableNameVal  
        key = itemKey  
        attributeUpdates = updatedValues  
    }
```

```
        }

        DynamoDbClient { region = "us-east-1" }.use { ddb ->
            ddb.updateItem(request)
            println("Item in $tableNameVal was updated")
        }
    }
```

- For API details, see [UpdateItem in AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
$rating = 0;
while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 || $rating > 10) {
    $rating = testable_readline("Rating (1-10): ");
}
$service->updateItemAttributeByKey($tableName, $key, 'rating', 'N', $rating);

public function updateItemAttributeByKey(
    string $tableName,
    array $key,
    string $attributeName,
    string $attributeType,
    string $newValue
) {
    $this->dynamoDbClient->updateItem([
        'Key' => $key['Item'],
        'TableName' => $tableName,
        'UpdateExpression' => "set #NV=:NV",
        'ExpressionAttributeNames' => [
            '#NV' => $attributeName,
        ],
        'ExpressionAttributeValues' => [
            ':NV' => [
                $attributeType => $newValue
            ]
        ],
    ]);
}
```

- For API details, see [UpdateItem in AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Update an item by using an update expression.

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None

    def update_movie(self, title, year, rating, plot):
        """
        Updates rating and plot data for a movie in the table.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating: The updated rating to give the movie.
        :param plot: The updated plot summary to give the movie.
        :return: The fields that were updated, with their new values.
        """
        try:
            response = self.table.update_item(
                Key={'year': year, 'title': title},
                UpdateExpression="set info.rating=:r, info.plot=:p",
                ExpressionAttributeValues={
                    ':r': Decimal(str(rating)),
                    ':p': plot,
                },
                ReturnValues="UPDATED_NEW")
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title, self.table.name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['Attributes']
```

Update an item by using an update expression that includes an arithmetic operation.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table

    def update_rating(self, title, year, rating_change):
        """
        Updates the quality rating of a movie in the table by using an arithmetic
        operation in the update expression. By specifying an arithmetic operation,
        you can adjust a value in a single request, rather than first getting its
        value and then setting its new value.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating_change: The amount to add to the current rating for the
        movie.
        :return: The updated rating.
        """
        try:
            response = self.table.update_item(
                Key={'year': year, 'title': title},
                UpdateExpression="set info.rating = info.rating + :val",
                ExpressionAttributeValues={':val': Decimal(str(rating_change))},
                ReturnValues="UPDATED_NEW")
        except ClientError as err:
```

```
logger.error(
    "Couldn't update movie %s in table %s. Here's why: %s: %s",
    title, self.table.name,
    err.response['Error']['Code'], err.response['Error']['Message'])
raise
else:
    return response['Attributes']
```

Update an item only when it meets certain conditions.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table

    def remove_actors(self, title, year, actor_threshold):
        """
        Removes an actor from a movie, but only when the number of actors is
        greater than a specified threshold. If the movie does not list more than the
        threshold, no actors are removed.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param actor_threshold: The threshold of actors to check.
        :return: The movie data after the update.
        """
        try:
            response = self.table.update_item(
                Key={'year': year, 'title': title},
                UpdateExpression="remove info.actors[0]",
                ConditionExpression="size(info.actors) > :num",
                ExpressionAttributeValues={':num': actor_threshold},
                ReturnValues="ALL_NEW")
        except ClientError as err:
            if err.response['Error']['Code'] == "ConditionalCheckFailedException":
                logger.warning(
                    "Didn't update %s because it has fewer than %s actors.",
                    title, actor_threshold + 1)
            else:
                logger.error(
                    "Couldn't update movie %s. Here's why: %s: %s",
                    title, err.response['Error']['Code'], err.response['Error']
                ['Message'])
                raise
        else:
            return response['Attributes']
```

- For API details, see [UpdateItem in AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Updates rating and plot data for a movie in the table.
#
```

```
# @param title [String] The title of the movie to update.
# @param year [Int] The release year of the movie to update.
# @param rating [Float] The updated rating to give the movie.
# @param plot [String] The updated plot summary to give the movie.
# @return [Hash] The fields that were updated, with their new values.
def update_movie(title:, year:, rating:, plot:)
    response = @table.update_item(
        key: {"year" => year, "title" => title},
        update_expression: "set info.rating=:r, info.plot=:p",
        expression_attribute_values: { ":r" => rating, ":p" => plot },
        return_values: "UPDATED_NEW")
    rescue Aws::Errors::ServiceError => e
        puts("Couldn't update movie #{title} in table #{@table.name}. Here's why:")
        puts("\t#{e.code}: #{e.message}")
        raise
    else
        response.attributes
    end
```

- For API details, see [UpdateItem in AWS SDK for Ruby API Reference](#).

## Write a batch of DynamoDB items using an AWS SDK

The following code examples show how to write a batch of DynamoDB items.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Writes a batch of items to the movie table.

```
/// <summary>
/// Loads the contents of a JSON file into a list of movies to be
/// added to the DynamoDB table.
/// </summary>
/// <param name="movieFileName">The full path to the JSON file.</param>
/// <returns>A generic list of movie objects.</returns>
public static List<Movie> ImportMovies(string movieFileName)
{
    if (!File.Exists(movieFileName))
    {
        return null;
    }

    using var sr = new StreamReader(movieFileName);
    string json = sr.ReadToEnd();
    var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

    // Now return the first 250 entries.
    return allMovies.GetRange(0, 250);
}

/// <summary>
/// Writes 250 items to the movie table.
/// </summary>
/// <param name="client">The initialized DynamoDB client object.</param>
/// <param name="movieFileName">A string containing the full path to
```

```
    ///> the JSON file containing movie data.</param>
    ///> <returns>A long integer value representing the number of movies
    ///> imported from the JSON file.</returns>
    public static async Task<long> BatchWriteItemsAsync(
        AmazonDynamoDBClient client,
        string movieFileName)
    {
        var movies = ImportMovies(movieFileName);
        if (movies is null)
        {
            Console.WriteLine("Couldn't find the JSON file with movie data.");
            return 0;
        }

        var context = new DynamoDBContext(client);

        var bookBatch = context.CreateBatchWrite<Movie>();
        bookBatch.AddPutItems(movies);

        Console.WriteLine("Adding imported movies to the table.");
        await bookBatch.ExecuteAsync();

        return movies.Count;
    }
```

- For API details, see [BatchWriteItem](#) in *AWS SDK for .NET API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified
// table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// AddMovieBatch adds a slice of movies to the DynamoDB table. The function sends
// batches of 25 movies to DynamoDB until all movies are added or it reaches the
// specified maximum.
func (basics TableBasics) AddMovieBatch(movies []Movie, maxMovies int) (int, error)
{
    var err error
    var item map[string]types.AttributeValue
    written := 0
    batchSize := 25 // DynamoDB allows a maximum batch size of 25 items.
    start := 0
    end := start + batchSize
    for start < maxMovies && start < len(movies) {
        var writeReqs []types.WriteRequest
        if end > len(movies) {
```

```
        end = len(movies)
    }
    for _, movie := range movies[start:end] {
        item, err = attributevalue.MarshalMap(movie)
        if err != nil {
            log.Printf("Couldn't marshal movie %v for batch writing. Here's why: %v\n",
movie.Title, err)
        } else {
            writeReqs = append(
                writeReqs,
                types.WriteRequest{PutRequest: &types.PutRequest{Item: item}},
            )
        }
    }
    _, err = basics.DynamoDbClient.BatchWriteItem(context.TODO(),
&dynamodb.BatchWriteItemInput{
    RequestItems: map[string][]types.WriteRequest{basics.TableName: writeReqs}})
    if err != nil {
        log.Printf("Couldn't add a batch of movies to %v. Here's why: %v\n",
basics.TableName, err)
    } else {
        written += len(writeReqs)
    }
    start = end
    end += batchSize
}

return written, err
}
```

- For API details, see [BatchWriteItem](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Inserts many items into a table by using the enhanced client.

```
public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {

    try {
        DynamoDbTable<Customer> customerMappedTable =
enhancedClient.table("Customer", TableSchema.fromBean(Customer.class));
        DynamoDbTable<Music> musicMappedTable = enhancedClient.table("Music",
TableSchema.fromBean(Music.class));
        LocalDate localDate = LocalDate.parse("2020-04-07");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        Customer record2 = new Customer();
        record2.setCustName("Fred Pink");
        record2.setId("id110");
        record2.setEmail("fredp@noserver.com");
        record2.setRegistrationDate(instant) ;

        Customer record3 = new Customer();
        record3.setCustName("Susan Pink");
        record3.setId("id120");
    }
}
```

```
record3.setEmail("spink@noserver.com");
record3.setRegistrationDate(instant) ;

Customer record4 = new Customer();
record4.setCustName("Jerry orange");
record4.setId("id101");
record4.setEmail("jorange@noserver.com");
record4.setRegistrationDate(instant) ;

BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest =
BatchWriteItemEnhancedRequest.builder()
    .writeBatches(
        WriteBatch.builder(Customer.class) // add
items to the Customer table
        .mappedTableResource(customerMappedTable)
        .addPutItem(builder -> builder.item(record2))
        .addPutItem(builder -> builder.item(record3))
        .addPutItem(builder -> builder.item(record4))
        .build(),
        WriteBatch.builder(Music.class) // delete
an item from the Music table
        .mappedTableResource(musicMappedTable)
        .addDeleteItem(builder -> builder.key(
            Key.builder().partitionValue("Famous
Band").build()))
        .build())
    .build();

// Add three items to the Customer table and delete one item from the
Music table
enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);

System.out.println("done");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [BatchWriteItem](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: true, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {
    marshallOptions,
    unmarshallOptions,
});

export { ddbDocClient };
```

Add the items to the table.

```
import fs from "fs";
import * as R from "ramda";
import { ddbDocClient } from "../libs/ddbDocClient.js";
import { BatchWriteCommand } from "@aws-sdk/lib-dynamodb";

export const writeData = async () => {
    // Before you run this example, download 'movies.json' from https://
    // docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Js.02.html,
    // and put it in the same folder as the example.
    // Get the movie data parse to convert into a JSON object.
    const allMovies = JSON.parse(fs.readFileSync("moviedata.json", "utf8"));
    // Split the table into segments of 25.
    const dataSegments = R.splitEvery(25, allMovies);
    const TABLE_NAME = "TABLE_NAME"
    try {
        // Loop batch write operation 10 times to upload 250 items.
        for (let i = 0; i < 10; i++) {
            const segment = dataSegments[i];
            for (let j = 0; j < 25; j++) {
                const params = {
                    RequestItems: {
                        [TABLE_NAME]: [
                            {
                                // Destination Amazon DynamoDB table name.
                                PutRequest: {
                                    Item: {
                                        year: segment[j].year,
                                        title: segment[j].title,
                                        info: segment[j].info,
                                    },
                                },
                            ],
                        ],
                    },
                };
            }
        }
    }
}
```

```
        ],
      },
    };
    ddbDocClient.send(new BatchWriteCommand(params));
  }
  console.log("Success, table updated.");
}
} catch (error) {
  console.log("Error", error);
}
};

writeData();

```

- For API details, see [BatchWriteItem](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
  RequestItems: [
    {
      "TABLE_NAME": [
        {
          PutRequest: {
            Item: {
              "KEY": { "N": "KEY_VALUE" },
              "ATTRIBUTE_1": { "S": "ATTRIBUTE_1_VALUE" },
              "ATTRIBUTE_2": { "N": "ATTRIBUTE_2_VALUE" }
            }
          }
        },
        {
          PutRequest: {
            Item: {
              "KEY": { "N": "KEY_VALUE" },
              "ATTRIBUTE_1": { "S": "ATTRIBUTE_1_VALUE" },
              "ATTRIBUTE_2": { "N": "ATTRIBUTE_2_VALUE" }
            }
          }
        }
      ]
    }
  ];
};

ddb.batchWriteItem(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [BatchWriteItem](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function writeBatch(string $TableName, array $Batch, int $depth = 2)
{
    if (--$depth <= 0) {
        throw new Exception("Max depth exceeded. Please try with fewer batch
items or increase depth.");
    }

    $marshal = new Marshaler();
    $total = 0;
    foreach (array_chunk($Batch, 25) as $Items) {
        foreach ($Items as $Item) {
            $BatchWrite['RequestItems'][$TableName][] = ['PutRequest' =>
['Item' => $marshal->marshalItem($Item)]];
        }
        try {
            echo "Batching another " . count($Items) . " for a total of " .
($total += count($Items)) . " items!\n";
            $response = $this->dynamoDbClient->batchWriteItem($BatchWrite);
            $BatchWrite = [];
        } catch (Exception $e) {
            echo "uh oh...";
            echo $e->getMessage();
            die();
        }
        if ($total >= 250) {
            echo "250 movies is probably enough. Right? We can stop there.\n";
            break;
        }
    }
}
```

- For API details, see [BatchWriteItem](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
```

```

        self.dyn_resource = dyn_resource
        self.table = None

    def write_batch(self, movies):
        """
        Fills an Amazon DynamoDB table with the specified data, using the Boto3
        Table.batch_writer() function to put the items in the table.
        Inside the context manager, Table.batch_writer builds a list of
        requests. On exiting the context manager, Table.batch_writer starts sending
        batches of write requests to Amazon DynamoDB and automatically
        handles chunking, buffering, and retrying.

        :param movies: The data to put in the table. Each item must contain at
        least
                           the keys required by the schema that was specified when the
                           table was created.
        """
        try:
            with self.table.batch_writer() as writer:
                for movie in movies:
                    writer.put_item(Item=movie)
        except ClientError as err:
            logger.error(
                "Couldn't load data into table %s. Here's why: %s: %s",
                self.table.name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise

```

- For API details, see [BatchWriteItem](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

# Fills an Amazon DynamoDB table with the specified data. Items are sent in
# batches of 25 until all items are written.
#
# @param movies [Enumerable] The data to put in the table. Each item must contain
at least
#                           the keys required by the schema that was specified
when the
#                           table was created.
#
def write_batch(movies)
  index = 0
  slice_size = 25
  while index < movies.length
    movie_items = []
    movies[index, slice_size].each do |movie|
      movie_items.append({put_request: { item: movie }})
    end
    @dynamo_resource.batch_write_item({request_items: { @table.name =>
movie_items }})
    index += slice_size
  end
rescue Aws::Errors::ServiceError => e
  puts(
    "Couldn't load data into table #{@table.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")

```

```
    raise
end
```

- For API details, see [BatchWriteItem](#) in *AWS SDK for Ruby API Reference*.

## Scenarios for DynamoDB using AWS SDKs

The following code examples show how to use Amazon DynamoDB with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Accelerate DynamoDB reads with DAX using an AWS SDK \(p. 449\)](#)
- [Get started using DynamoDB tables, items, and queries using an AWS SDK \(p. 454\)](#)
- [Query a DynamoDB table by using batches of PartiQL statements and an AWS SDK \(p. 522\)](#)
- [Query a DynamoDB table using PartiQL and an AWS SDK \(p. 561\)](#)

## Accelerate DynamoDB reads with DAX using an AWS SDK

The following code example shows how to:

- Create and write data to a table with both the DAX and SDK clients.
- Get, query, and scan the table with both the DAX and SDK clients and compare their performance.

For more information, see [Developing with the DynamoDB Accelerator Client](#).

### Python

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a table with either the DAX or Boto3 client.

```
import boto3

def create_dax_table(dyn_resource=None):
    """
    Creates a DynamoDB table.

    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The newly created table.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource('dynamodb')

    table_name = 'TryDaxTable'
    params = {
        'TableName': table_name,
        'KeySchema': [
            {'AttributeName': 'partition_key', 'KeyType': 'HASH'},
            {'AttributeName': 'sort_key', 'KeyType': 'RANGE'}
        ],
        'AttributeDefinitions': [
```

```
        {'AttributeName': 'partition_key', 'AttributeType': 'N'},
        {'AttributeName': 'sort_key', 'AttributeType': 'N'}
    ],
    'ProvisionedThroughput': {
        'ReadCapacityUnits': 10,
        'WriteCapacityUnits': 10
    }
}
table = dyn_resource.create_table(**params)
print(f"Creating {table_name}...")
table.wait_until_exists()
return table

if __name__ == '__main__':
    dax_table = create_dax_table()
    print(f"Created table.")
```

Write test data to the table.

```
import boto3

def write_data_to_dax_table(key_count, item_size, dyn_resource=None):
    """
    Writes test data to the demonstration table.

    :param key_count: The number of partition and sort keys to use to populate the
                      table. The total number of items is key_count * key_count.
    :param item_size: The size of non-key data for each test item.
    :param dyn_resource: Either a Boto3 or DAX resource.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource('dynamodb')

    table = dyn_resource.Table('TryDaxTable')
    some_data = 'X' * item_size

    for partition_key in range(1, key_count + 1):
        for sort_key in range(1, key_count + 1):
            table.put_item(Item={
                'partition_key': partition_key,
                'sort_key': sort_key,
                'some_data': some_data
            })
    print(f"Put item ({partition_key}, {sort_key}) succeeded.")

if __name__ == '__main__':
    write_key_count = 10
    write_item_size = 1000
    print(f"Writing {write_key_count}*{write_key_count} items to the table. "
          f"Each item is {write_item_size} characters.")
    write_data_to_dax_table(write_key_count, write_item_size)
```

Get items for a number of iterations for both the DAX client and the Boto3 client and report the time spent for each.

```
import argparse
import sys
import time
```

```
import amazondax
import boto3

def get_item_test(key_count, iterations, dyn_resource=None):
    """
    Gets items from the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param key_count: The number of items to get from the table in each iteration.
    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource('dynamodb')

    table = dyn_resource.Table('TryDaxTable')
    start = time.perf_counter()
    for _ in range(iterations):
        for partition_key in range(1, key_count + 1):
            for sort_key in range(1, key_count + 1):
                table.get_item(Key={
                    'partition_key': partition_key,
                    'sort_key': sort_key
                })
                print('.', end='')
                sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end

if __name__ == '__main__':
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        'endpoint_url', nargs='?',
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
used.")
    args = parser.parse_args()

    test_key_count = 10
    test_iterations = 50
    if args.endpoint_url:
        print(f"Getting each item from the table {test_iterations} times, "
              f"using the DAX client.")
        # Use a with statement so the DAX client closes the cluster after
        completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as
dax:
            test_start, test_end = get_item_test(
                test_key_count, test_iterations, dyn_resource=dax)
    else:
        print(f"Getting each item from the table {test_iterations} times, "
              f"using the Boto3 client.")
        test_start, test_end = get_item_test(
            test_key_count, test_iterations)
    print(f"Total time: {test_end - test_start:.4f} sec. Average time: "
          f"{{(test_end - test_start)/ test_iterations}}")
```

Query the table for a number of iterations for both the DAX client and the Boto3 client and report the time spent for each.

```
import argparse
import time
import sys
import amazondax
import boto3
from boto3.dynamodb.conditions import Key

def query_test(partition_key, sort_keys, iterations, dyn_resource=None):
    """
    Queries the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param partition_key: The partition key value to use in the query. The query
                          returns items that have partition keys equal to this
                          value.
    :param sort_keys: The range of sort key values for the query. The query returns
                     items that have sort key values between these two values.
    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource('dynamodb')

    table = dyn_resource.Table('TryDaxTable')
    key_condition_expression = \
        Key('partition_key').eq(partition_key) & \
        Key('sort_key').between(*sort_keys)

    start = time.perf_counter()
    for _ in range(iterations):
        table.query(KeyConditionExpression=key_condition_expression)
        print('.', end='')
        sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end

if __name__ == '__main__':
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        'endpoint_url', nargs='?',
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
used.")
    args = parser.parse_args()

    test_partition_key = 5
    test_sort_keys = (2, 9)
    test_iterations = 100
    if args.endpoint_url:
        print(f"Querying the table {test_iterations} times, using the DAX client.")
        # Use a with statement so the DAX client closes the cluster after
        completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as
dax:
            test_start, test_end = query_test(
                test_partition_key, test_sort_keys, test_iterations,
                dyn_resource=dax)
        else:
            print(f"Querying the table {test_iterations} times, using the Boto3
client.")
```

```
    test_start, test_end = query_test(
        test_partition_key, test_sort_keys, test_iterations)

    print(f"Total time: {test_end - test_start:.4f} sec. Average time: "
        f"{{(test_end - test_start)/test_iterations}}")
```

Scan the table for a number of iterations for both the DAX client and the Boto3 client and report the time spent for each.

```
import argparse
import time
import sys
import amazondax
import boto3

def scan_test(iterations, dyn_resource=None):
    """
    Scans the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource('dynamodb')

    table = dyn_resource.Table('TryDaxTable')
    start = time.perf_counter()
    for _ in range(iterations):
        table.scan()
        print('.', end='')
        sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end

if __name__ == '__main__':
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        'endpoint_url', nargs='?',
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not
used.")
    args = parser.parse_args()

    test_iterations = 100
    if args.endpoint_url:
        print(f"Scanning the table {test_iterations} times, using the DAX client.")
        # Use a with statement so the DAX client closes the cluster after
completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as
dax:
            test_start, test_end = scan_test(test_iterations, dyn_resource=dax)
    else:
        print(f"Scanning the table {test_iterations} times, using the Boto3
client.")
        test_start, test_end = scan_test(test_iterations)
    print(f"Total time: {test_end - test_start:.4f} sec. Average time: "
        f"{{(test_end - test_start)/test_iterations}}")
```

Delete the table.

```
import boto3

def delete_dax_table(dyn_resource=None):
    """
    Deletes the demonstration table.

    :param dyn_resource: Either a Boto3 or DAX resource.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource('dynamodb')

    table = dyn_resource.Table('TryDaxTable')
    table.delete()

    print(f"Deleting {table.name}...")
    table.wait_until_not_exists()

if __name__ == '__main__':
    delete_dax_table()
    print("Table deleted!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateTable](#)
  - [DeleteTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)

## Get started using DynamoDB tables, items, and queries using an AWS SDK

The following code examples show how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.
- Delete a movie from the table.
- Delete the table.

.NET

**AWS SDK for .NET**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// This example application performs the following basic Amazon DynamoDB
// functions:
//
//      CreateTableAsync
//      PutItemAsync
//      UpdateItemAsync
//      BatchWriteItemAsync
//      GetItemAsync
//      DeleteItemAsync
//      Query
//      Scan
//      DeleteItemAsync
//
// The code in this example uses the AWS SDK for .NET version 3.7 and .NET 5.
// Before you run this example, download 'movies.json' from
// https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/
// sample_files,
// and put it in the same folder as the example.
namespace DynamoDB_Basics_Scenario
{
    public class DynamoDB_Basics
    {
        // Separator for the console display.
        private static readonly string SepBar = new string('-', 80);

        public static async Task Main()
        {
            var client = new AmazonDynamoDBClient();

            var tableName = "movie_table";

            // relative path to moviedata.json in the local repository.
            var movieFileName = @"..\..\..\..\..\..\..\resources\sample_files
\\movies.json";

            DisplayInstructions();

            // Create a new table and wait for it to be active.
            Console.WriteLine($"Creating the new table: {tableName}");

            var success = await DynamoDbMethods.CreateMovieTableAsync(client,
            tableName);

            if (success)
            {
                Console.WriteLine($"\\nTable: {tableName} successfully created.");
            }
            else
            {
                Console.WriteLine($"\\nCould not create {tableName}.");
            }

            WaitForEnter();

            // Add a single new movie to the table.
            var newMovie = new Movie
```

```
{  
    Year = 2021,  
    Title = "Spider-Man: No Way Home",  
};  
  
success = await DynamoDbMethods.PutItemAsync(client, newMovie,  
tableName);  
if (success)  
{  
    Console.WriteLine($"Added {newMovie.Title} to the table.");  
}  
else  
{  
    Console.WriteLine("Could not add movie to table.");  
}  
  
WaitForEnter();  
  
// Update the new movie by adding a plot and rank.  
var newInfo = new MovieInfo  
{  
    Plot = "With Spider-Man's identity now revealed, Peter asks" +  
        "Doctor Strange for help. When a spell goes wrong, dangerous" +  
        "foes from other worlds start to appear, forcing Peter to" +  
        "discover what it truly means to be Spider-Man.",  
    Rank = 9,  
};  
  
success = await DynamoDbMethods.UpdateItemAsync(client, newMovie,  
newInfo, tableName);  
if (success)  
{  
    Console.WriteLine($"Successfully updated the movie:  
{newMovie.Title}");  
}  
else  
{  
    Console.WriteLine("Could not update the movie.");  
}  
  
WaitForEnter();  
  
// Add a batch of movies to the DynamoDB table from a list of  
// movies in a JSON file.  
var itemCount = await DynamoDbMethods.BatchWriteItemsAsync(client,  
movieFileName);  
Console.WriteLine($"Added {itemCount} movies to the table.");  
  
WaitForEnter();  
  
// Get a movie by key. (partition + sort)  
var lookupMovie = new Movie  
{  
    Title = "Jurassic Park",  
    Year = 1993,  
};  
  
Console.WriteLine("Looking for the movie \"Jurassic Park\".");  
var item = await DynamoDbMethods.GetItemAsync(client, lookupMovie,  
tableName);  
if (item.Count > 0)  
{  
    DynamoDbMethods.DisplayItem(item);  
}  
else  
{
```

```
        Console.WriteLine($"Couldn't find {lookupMovie.Title}");
    }

    WaitForEnter();

    // Delete a movie.
    var movieToDelete = new Movie
    {
        Title = "The Town",
        Year = 2010,
    };

    success = await DynamoDbMethods.DeleteItemAsync(client, tableName,
movieToDelete);

    if (success)
    {
        Console.WriteLine($"Successfully deleted {movieToDelete.Title}.");
    }
    else
    {
        Console.WriteLine($"Could not delete {movieToDelete.Title}.");
    }

    WaitForEnter();

    // Use Query to find all the movies released in 2010.
    int findYear = 2010;
    Console.WriteLine($"Movies released in {findYear}");
    var queryCount = await DynamoDbMethods.QueryMoviesAsync(client,
tableName, findYear);
    Console.WriteLine($"Found {queryCount} movies released in {findYear}");

    WaitForEnter();

    // Use Scan to get a list of movies from 2001 to 2011.
    int startYear = 2001;
    int endYear = 2011;
    var scanCount = await DynamoDbMethods.ScanTableAsync(client, tableName,
startYear, endYear);
    Console.WriteLine($"Found {scanCount} movies released between
{startYear} and {endYear}");

    WaitForEnter();

    // Delete the table.
    success = await DynamoDbMethods.DeleteTableAsync(client, tableName);

    if (success)
    {
        Console.WriteLine($"Successfully deleted {tableName}");
    }
    else
    {
        Console.WriteLine($"Could not delete {tableName}");
    }

    Console.WriteLine("The DynamoDB Basics example application is done.");

    WaitForEnter();
}

/// <summary>
/// Displays the description of the application on the console.
/// </summary>
private static void DisplayInstructions()
```

```

    {
        Console.Clear();
        Console.WriteLine();
        Console.Write(new string(' ', 28));
        Console.WriteLine("DynamoDB Basics Example");
        Console.WriteLine(SepBar);
        Console.WriteLine("This demo application shows the basics of using
DynamoDB with the AWS SDK for");
        Console.WriteLine(".NET version 3.7 and .NET Core 5.");
        Console.WriteLine(SepBar);
        Console.WriteLine("The application does the following:");
        Console.WriteLine("\t1. Creates a table with partition: year and
sort:title.");
        Console.WriteLine("\t2. Adds a single movie to the table.");
        Console.WriteLine("\t3. Adds movies to the table from
moviedata.json.");
        Console.WriteLine("\t4. Updates the rating and plot of the movie that
was just added.");
        Console.WriteLine("\t5. Gets a movie using its key (partition +
sort).");
        Console.WriteLine("\t6. Deletes a movie.");
        Console.WriteLine("\t7. Uses QueryAsync to return all movies released
in a given year.");
        Console.WriteLine("\t8. Uses ScanAsync to return all movies released
within a range of years.");
        Console.WriteLine("\t9. Finally, it deletes the table that was just
created.");
        WaitForEnter();
    }

    ///<summary>
    /// Simple method to wait for the Enter key to be pressed.
    ///</summary>
    private static void WaitForEnter()
    {
        Console.WriteLine("\nPress <Enter> to continue.");
        Console.WriteLine(SepBar);
        _ = Console.ReadLine();
    }
}

```

Creates a table to contain movie data.

```

    ///<summary>
    /// Creates a new Amazon DynamoDB table and then waits for the new
    /// table to become active.
    ///</summary>
    ///<param name="client">An initialized Amazon DynamoDB client object.</
param>
    ///<param name="tableName">The name of the table to create.</param>
    ///<returns>A Boolean value indicating the success of the operation.</
returns>
    public static async Task<bool> CreateMovieTableAsync(AmazonDynamoDBClient
client, string tableName)
    {
        var response = await client.CreateTableAsync(new CreateTableRequest
        {
            TableName = tableName,
            AttributeDefinitions = new List<AttributeDefinition>()
            {
                new AttributeDefinition

```

```
{  
    AttributeName = "title",  
    AttributeType = "S",  
,  
    new AttributeDefinition  
{  
        AttributeName = "year",  
        AttributeType = "N",  
,  
},  
KeySchema = new List<KeySchemaElement>()  
{  
    new KeySchemaElement  
{  
        AttributeName = "year",  
        KeyType = "HASH",  
,  
    new KeySchemaElement  
{  
        AttributeName = "title",  
        KeyType = "RANGE",  
,  
},  
ProvisionedThroughput = new ProvisionedThroughput  
{  
    ReadCapacityUnits = 5,  
    WriteCapacityUnits = 5,  
,  
});  
  
// Wait until the table is ACTIVE and then report success.  
Console.WriteLine("Waiting for table to become active...");  
  
var request = new DescribeTableRequest  
{  
    TableName = response.TableDescription.TableName,  
};  
  
TableStatus status;  
  
int sleepDuration = 2000;  
  
do  
{  
    System.Threading.Thread.Sleep(sleepDuration);  
  
    var describeTableResponse = await  
client.DescribeTableAsync(request);  
    status = describeTableResponse.Table.TableStatus;  
  
    Console.WriteLine(".");  
}  
while (status != "ACTIVE");  
  
return status == TableStatus.ACTIVE;  
}
```

Adds a single movie to the table.

```
/// <summary>  
/// Adds a new item to the table.  
/// </summary>
```

```

/// <param name="client">An initialized Amazon DynamoDB client object.</param>
/// <param name="newMovie">A Movie object containing information for
/// the movie to add to the table.</param>
/// <param name="tableName">The name of the table where the item will be
added.</param>
/// <returns>A Boolean value that indicates the results of adding the
item.</returns>
public static async Task<bool> PutItemAsync(AmazonDynamoDBClient client,
Movie newMovie, string tableName)
{
    var item = new Dictionary<string, AttributeValue>
    {
        ["title"] = new AttributeValue { S = newMovie.Title },
        ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
    };

    var request = new PutItemRequest
    {
        TableName = tableName,
        Item = item,
    };

    var response = await client.PutItemAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

```

Updates a single item in a table.

```

/// <summary>
/// Updates an existing item in the movies table.
/// </summary>
/// <param name="client">An initialized Amazon DynamoDB client object.</param>
/// <param name="newMovie">A Movie object containing information for
/// the movie to update.</param>
/// <param name="newInfo">A MovieInfo object that contains the
/// information that will be changed.</param>
/// <param name="tableName">The name of the table that contains the
movie.</param>
/// <returns>A Boolean value that indicates the success of the operation.</returns>
public static async Task<bool> UpdateItemAsync(
    AmazonDynamoDBClient client,
    Movie newMovie,
    MovieInfo newInfo,
    string tableName)
{
    var key = new Dictionary<string, AttributeValue>
    {
        ["title"] = new AttributeValue { S = newMovie.Title },
        ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
    };
    var updates = new Dictionary<string, AttributeValueUpdate>
    {
        ["info.plot"] = new AttributeValueUpdate
        {
            Action = AttributeAction.PUT,
            Value = new AttributeValue { S = newInfo.Plot },
        },
        ["info.rating"] = new AttributeValueUpdate
    };
}

```

```

    {
        Action = AttributeAction.PUT,
        Value = new AttributeValue { N = newInfo.Rank.ToString() },
    },
};

var request = new UpdateItemRequest
{
    AttributeUpdates = updates,
    Key = key,
    TableName = tableName,
};

var response = await client.UpdateItemAsync(request);

return response.HttpStatusCode == System.Net HttpStatusCode.OK;
}

```

Retrieves a single item from the movie table.

```

///<summary>
/// Gets information about an existing movie from the table.
///</summary>
///<param name="client">An initialized Amazon DynamoDB client object.</param>
///<param name="newMovie">A Movie object containing information about
/// the movie to retrieve.</param>
///<param name="tableName">The name of the table containing the movie.</param>
///<returns>A Dictionary object containing information about the item
/// retrieved.</returns>
public static async Task<Dictionary<string, AttributeValue>>
GetItemAsync(AmazonDynamoDBClient client, Movie newMovie, string tableName)
{
    var key = new Dictionary<string, AttributeValue>
    {
        ["title"] = new AttributeValue { S = newMovie.Title },
        ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
    };

    var request = new GetItemRequest
    {
        Key = key,
        TableName = tableName,
    };

    var response = await client.GetItemAsync(request);
    return response.Item;
}

```

Writes a batch of items to the movie table.

```

///<summary>
/// Loads the contents of a JSON file into a list of movies to be
/// added to the DynamoDB table.
///</summary>
///<param name="movieFileName">The full path to the JSON file.</param>

```

```
/// <returns>A generic list of movie objects.</returns>
public static List<Movie> ImportMovies(string movieFileName)
{
    if (!File.Exists(movieFileName))
    {
        return null;
    }

    using var sr = new StreamReader(movieFileName);
    string json = sr.ReadToEnd();
    var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

    // Now return the first 250 entries.
    return allMovies.GetRange(0, 250);
}

/// <summary>
/// Writes 250 items to the movie table.
/// </summary>
/// <param name="client">The initialized DynamoDB client object.</param>
/// <param name="movieFileName">A string containing the full path to
/// the JSON file containing movie data.</param>
/// <returns>A long integer value representing the number of movies
/// imported from the JSON file.</returns>
public static async Task<long> BatchWriteItemsAsync(
    AmazonDynamoDBClient client,
    string movieFileName)
{
    var movies = ImportMovies(movieFileName);
    if (movies is null)
    {
        Console.WriteLine("Couldn't find the JSON file with movie data.");
        return 0;
    }

    var context = new DynamoDBContext(client);

    var bookBatch = context.CreateBatchWrite<Movie>();
    bookBatch.AddPutItems(movies);

    Console.WriteLine("Adding imported movies to the table.");
    await bookBatch.ExecuteAsync();

    return movies.Count;
}
```

Deletes a single item from the table.

```
/// <summary>
/// Deletes a single item from a DynamoDB table.
/// </summary>
/// <param name="client">The initialized DynamoDB client object.</param>
/// <param name="tableName">The name of the table from which the item
/// will be deleted.</param>
/// <param name="movieToDelete">A movie object containing the title and
/// year of the movie to delete.</param>
/// <returns>A Boolean value indicating the success or failure of the
/// delete operation.</returns>
public static async Task<bool> DeleteItemAsync(
    AmazonDynamoDBClient client,
    string tableName,
    Movie movieToDelete)
```

```
{  
    var key = new Dictionary<string, AttributeValue>  
    {  
        ["title"] = new AttributeValue { S = movieToDelete.Title },  
        ["year"] = new AttributeValue { N =  
            movieToDelete.Year.ToString() },  
    };  
  
    var request = new DeleteItemRequest  
    {  
        TableName = tableName,  
        Key = key,  
    };  
  
    var response = await client.DeleteItemAsync(request);  
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;  
}
```

Queries the table for movies released in a particular year.

```
/// <summary>  
/// Queries the table for movies released in a particular year and  
/// then displays the information for the movies returned.  
/// </summary>  
/// <param name="client">The initialized DynamoDB client object.</param>  
/// <param name="tableName">The name of the table to query.</param>  
/// <param name="year">The release year for which we want to  
/// view movies.</param>  
/// <returns>The number of movies that match the query.</returns>  
public static async Task<int> QueryMoviesAsync(AmazonDynamoDBClient client,  
string tableName, int year)  
{  
    var movieTable = Table.LoadTable(client, tableName);  
    var filter = new QueryFilter("year", QueryOperator.Equal, year);  
  
    Console.WriteLine("\nFind movies released in: {year}:");  
  
    var config = new QueryOperationConfig()  
    {  
        Limit = 10, // 10 items per page.  
        Select = SelectValues.SpecificAttributes,  
        AttributesToGet = new List<string>  
        {  
            "title",  
            "year",  
        },  
        ConsistentRead = true,  
        Filter = filter,  
    };  
  
    // Value used to track how many movies match the  
    // supplied criteria.  
    var moviesFound = 0;  
  
    Search search = movieTable.Query(config);  
    do  
    {  
        var movieList = await search.GetNextSetAsync();  
        moviesFound += movieList.Count;  
  
        foreach (var movie in movieList)  
        {
```

```
        DisplayDocument(movie);
    }
}
while (!search.IsDone);

return moviesFound;
}
```

Scans the table for movies released in a range of years.

```
public static async Task<int> ScanTableAsync(
    AmazonDynamoDBClient client,
    string tableName,
    int startYear,
    int endYear)
{
    var request = new ScanRequest
    {
        TableName = tableName,
        ExpressionAttributeNames = new Dictionary<string, string>
        {
            { "#yr", "year" },
        },
        ExpressionAttributeValues = new Dictionary<string, AttributeValue>
        {
            { ":y_a", new AttributeValue { N = startYear.ToString() } },
            { ":y_z", new AttributeValue { N = endYear.ToString() } },
        },
        FilterExpression = "#yr between :y_a and :y_z",
        ProjectionExpression = "#yr, title, info.actors[0], info.directors,
info.running_time_secs",
    };

    // Keep track of how many movies were found.
    int foundCount = 0;

    var response = new ScanResponse();
    do
    {
        response = await client.ScanAsync(request);
        foundCount += response.Items.Count;
        response.Items.ForEach(i => DisplayItem(i));
    }
    while (response.LastEvaluatedKey.Count > 1);
    return foundCount;
}
```

Deletes the movie table.

```
public static async Task<bool> DeleteTableAsync(AmazonDynamoDBClient
client, string tableName)
{
    var request = new DeleteTableRequest
    {
        TableName = tableName,
    };

    var response = await client.DeleteTableAsync(request);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
```

```
        Console.WriteLine($"Table {response.TableDescription.TableName}  
successfully deleted.");  
        return true;  
    }  
    else  
    {  
        Console.WriteLine("Could not delete table.");  
        return false;  
    }  
}
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.

- [BatchWriteItem](#)
- [CreateTable](#)
- [DeleteItem](#)
- [DeleteTable](#)
- [DescribeTable](#)
- [GetItem](#)
- [PutItem](#)
- [Query](#)
- [Scan](#)
- [UpdateItem](#)

C++

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
{  
    Aws::Client::ClientConfiguration clientConfig;  
    // 1. Create a table with partition: year (N) and sort: title (S).  
(CreateTable)  
    if  
(AwsDoc::DynamoDB::createDynamoDBTable(AwsDoc::DynamoDB::MOVIE_TABLE_NAME,  
                                         clientConfig)) {  
  
    AwsDoc::DynamoDB::dynamodbGettingStartedScenario(clientConfig);  
  
    // 9. Delete the table. (DeleteTable)  
    AwsDoc::DynamoDB::deleteDynamoTable(AwsDoc::DynamoDB::MOVIE_TABLE_NAME,  
                                         clientConfig);  
}  
}  
  
//! Scenario to modify and query a DynamoDB table.  
/*!  
 \sa dynamodbGettingStartedScenario()  
 \param clientConfiguration: Aws client configuration.  
 \return bool: Function succeeded.  
 */  
bool AwsDoc::DynamoDB::dynamodbGettingStartedScenario(  
    const Aws::Client::ClientConfiguration &clientConfiguration) {  
    std::cout << std::setfill('*') << std::setw(ASTERIX_FILL_WIDTH) << " " <<  
    std::endl;
```

```

        std::cout << "Welcome to the Amazon DynamoDB getting started demo." <<
        std::endl;
        std::cout << std::setfill('*') << std::setw(ASTERIX_FILL_WIDTH) << " " <<
        std::endl;

        Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

        // 2. Add a new movie.
        Aws::String title;
        float rating;
        int year;
        Aws::String plot;
    {
        title = askQuestion(
            "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate
it? ",
                                         1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::PutItemRequest putItemRequest;
        putItemRequest.SetTableName(MOVIE_TABLE_NAME);

        putItemRequest.AddItem(YEAR_KEY,
                               Aws::DynamoDB::Model::AttributeValue().SetN(year));
        putItemRequest.AddItem(TITLE_KEY,
                               Aws::DynamoDB::Model::AttributeValue().SetS(title));

        // Create attribute for the info map.
        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);

        putItemRequest.AddItem(INFO_KEY, infoMapAttribute);

        Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
            putItemRequest);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add an item: " <<
            outcome.GetError().GetMessage()
            << std::endl;
            return false;
        }
    }

    std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
    << std::endl;

    // 3. Update the rating and plot of the movie by using an update expression.
    {
        rating = askQuestionForFloatRange(
            Aws::String("\nLet's update your movie.\nYou rated it " +
            std::to_string(rating)
            + ", what new rating would you give it? ", 1, 10);
        plot = askQuestion(Aws::String("You summarized the plot as ''") + plot +

```

```

        "'.\nWhat would you say now? ");

Aws::DynamoDB::Model::UpdateItemRequest request;
request.SetTableName(MOVIE_TABLE_NAME);
request.AddKey(TITLE_KEY,
Aws::DynamoDB::Model::AttributeValue().SetS(title));
request.AddKey(YEAR_KEY,
Aws::DynamoDB::Model::AttributeValue().SetN(year));
std::stringstream expressionStream;
expressionStream << "set " << INFO_KEY << "." << RATING_KEY << " =:r, "
<< INFO_KEY << "." << PLOT_KEY << " =:p";
request.SetUpdateExpression(expressionStream.str());
request.SetExpressionAttributeValues({
                                            {"":r",
Aws::DynamoDB::Model::AttributeValue().SetN(
                                                rating)},
                                            {"":p",
Aws::DynamoDB::Model::AttributeValue().SetS(
                                                plot)}
                                            });

request.SetReturnValues(Aws::DynamoDB::Model::ReturnValue::UPDATED_NEW);

const Aws::DynamoDB::Model::UpdateItemOutcome &result =
dynamoClient.UpdateItem(
    request);
if (!result.IsSuccess()) {
    std::cerr << "Error updating movie " + result.GetError().GetMessage()
        << std::endl;
    return false;
}
}

std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;

// 4. Put 250 movies in the table from moviedata.json.
{
    std::cout << "Adding movies from a json file to the database." <<
std::endl;
    const size_t MAX_SIZE_FOR_BATCH_WRITE = 25;
    const size_t MOVIES_TO_WRITE = 10 * MAX_SIZE_FOR_BATCH_WRITE;
    Aws::String jsonString = getMovieJSON();
    if (!jsonString.empty()) {
        Aws::Utils::JsonValue json(jsonString);
        Aws::Utils::Array<Aws::Utils::Json::JsonView> movieJsons =
json.View().AsArray();
        Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;

        // To add movies with a cross-section of years, use an appropriate
increment
        // value for iterating through the database.
        size_t increment = movieJsons.GetLength() / MOVIES_TO_WRITE;
        for (size_t i = 0; i < movieJsons.GetLength(); i += increment) {
            writeRequests.push_back(Aws::DynamoDB::Model::WriteRequest());
            Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
putItems = movieJsonViewToAttributeMap(
                movieJsons[i]);
            Aws::DynamoDB::Model::PutRequest putRequest;
            putRequest.SetItem(putItems);
            writeRequests.back().SetPutRequest(putRequest);
            if (writeRequests.size() == MAX_SIZE_FOR_BATCH_WRITE) {
                Aws::DynamoDB::Model::BatchWriteItemRequest request;
                request.AddRequestItems(MOVIE_TABLE_NAME, writeRequests);
                const Aws::DynamoDB::Model::BatchWriteItemOutcome &outcome =
dynamoClient.BatchWriteItem(
                    request);

```

```

        if (!outcome.IsSuccess()) {
            std::cerr << "Unable to batch write movie data: "
                << outcome.GetError().GetMessage()
                << std::endl;
            writeRequests.clear();
            break;
        }
        else {
            std::cout << "Added batch of " << writeRequests.size()
                << " movies to the database."
                << std::endl;
        }
        writeRequests.clear();
    }
}

std::cout << std::setfill('*') << std::setw(ASTERIX_FILL_WIDTH) << " " <<
std::endl;

// 5. Get a movie by Key (partition + sort).
{
    Aws::String titleToGet("King Kong");
    Aws::String answer = askQuestion(Aws::String(
        "Let's move on...Would you like to get info about '" + titleToGet +
        "'? (y/n) "));
    if (answer == "y") {
        Aws::DynamoDB::Model::GetItemRequest request;
        request.SetTableName(MOVIE_TABLE_NAME);
        request.AddKey(TITLE_KEY,
                    Aws::DynamoDB::Model::AttributeValue().SetS(titleToGet));
        request.AddKey(YEAR_KEY,
                    Aws::DynamoDB::Model::AttributeValue().SetN(1933));

        const Aws::DynamoDB::Model::GetItemOutcome &result =
dynamoClient.GetItem(
            request);
        if (!result.IsSuccess()) {
            std::cerr << "Error " << result.GetError().GetMessage();
        }
        else {
            const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = result.GetResult().GetItem();
            if (!item.empty()) {
                std::cout << "\nHere's what I found:" << std::endl;
                printMovieInfo(item);
            }
            else {
                std::cout << "\nThe movie was not found in the database."
                    << std::endl;
            }
        }
    }
}

// 6. Use Query with a key condition expression to return all movies
//     released in a given year.
Aws::String doAgain = "n";
do {
    Aws::DynamoDB::Model::QueryRequest req;
    req.SetTableName(MOVIE_TABLE_NAME);
    // "year" is a DynamoDB reserved keyword and must be replaced with an

```

```

// expression attribute name.
req.SetKeyConditionExpression("#dynobase_year = :valueToMatch");
req.SetExpressionAttributeNames({{"#dynobase_year", YEAR_KEY}});

int yearToMatch = askQuestionForIntRange(
    "\nLet's get a list of movies released in"
    " a given year. Enter a year between 1972 and 2018 ",
    1972, 2018);
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributeValues;
attributeValues.emplace(":valueToMatch",
                      Aws::DynamoDB::Model::AttributeValue().SetN(
                          yearToMatch));
req.SetExpressionAttributeValues(attributeValues);

const Aws::DynamoDB::Model::QueryOutcome &result = dynamoClient.Query(req);
if (result.IsSuccess()) {
    const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
    if (!items.empty()) {
        std::cout << "\nThere were " << items.size()
            << " movies in the database from "
            << yearToMatch << "." << std::endl;
        for (const auto &item: items) {
            printMovieInfo(item);
        }
        doAgain = "n";
    }
    else {
        std::cout << "\nNo movies from " << yearToMatch
            << " were found in the database"
            << std::endl;
        doAgain = askQuestion(Aws::String("Try another year? (y/n) "));
    }
}
else {
    std::cerr << "Failed to Query items: " <<
result.GetError().GetMessage()
            << std::endl;
}

} while (doAgain == "y");

// 7. Use Scan to return movies released within a range of years.
//     Show how to paginate data using ExclusiveStartKey. (Scan +
FilterExpression)
{
    int startYear = askQuestionForIntRange("\nNow let's scan a range of years "
                                         "for movies in the database. Enter a
start year: ",
                                         1972, 2018);
    int endYear = askQuestionForIntRange("\nEnter an end year: ",
                                         startYear, 2018);
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
exclusiveStartKey;
    do {
        Aws::DynamoDB::Model::ScanRequest scanRequest;
        scanRequest.SetTableName(MOVIE_TABLE_NAME);
        scanRequest.SetFilterExpression(
            "#dynobase_year >= :startYear AND #dynobase_year <= :endYear");
        scanRequest.SetExpressionAttributeNames({{"#dynobase_year",
YEAR_KEY}});
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributeValues;
        attributeValues.emplace(":startYear",

```

```

        Aws::DynamoDB::Model::AttributeValue().SetN(
            startYear));
    attributeValues.emplace(":endYear",
        Aws::DynamoDB::Model::AttributeValue().SetN(
            endYear));
    scanRequest.SetExpressionAttributeValues(attributeValues);

    if (!exclusiveStartKey.empty()) {
        scanRequest.SetExclusiveStartKey(exclusiveStartKey);
    }

    const Aws::DynamoDB::Model::ScanOutcome &result = dynamoClient.Scan(
        scanRequest);
    if (result.IsSuccess()) {
        const Aws::Vector<Aws::Map<Aws::String,
        Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
        if (!items.empty()) {
            std::stringstream stringStream;
            stringStream << "\nFound " << items.size() << " movies in one
scan."
                << " How many would you like to see? ";
            size_t count = askQuestionForInt(stringStream.str());
            for (size_t i = 0; i < count && i < items.size(); ++i) {
                printMovieInfo(items[i]);
            }
        }
        else {
            std::cout << "\nNo movies in the database between " <<
startYear <<
                " and " << endYear << "." << std::endl;
        }

        exclusiveStartKey = result.GetResult().GetLastEvaluatedKey();
        if (!exclusiveStartKey.empty()) {
            std::cout << "Not all movies were retrieved. Scanning for
more."
                << std::endl;
        }
        else {
            std::cout << "All movies were retrieved with this scan."
                << std::endl;
        }
    }
    else {
        std::cerr << "Failed to Scan movies: "
            << result.GetError().GetMessage() << std::endl;
    }
} while (!exclusiveStartKey.empty());
}

// 8. Delete a movie. (DeleteItem)
{
    std::stringstream stringStream;
    stringStream << "\nWould you like to delete the movie " << title
        << " from the database? (y/n) ";
    Aws::String answer = askQuestion(stringStream.str());
    if (answer == "y") {
        Aws::DynamoDB::Model::DeleteItemRequest request;
        request.AddKey(YEAR_KEY,
        Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.AddKey(TITLE_KEY,
            Aws::DynamoDB::Model::AttributeValue().SetS(title));
        request.SetTableName(MOVIE_TABLE_NAME);

        const Aws::DynamoDB::Model::DeleteItemOutcome &result =
dynamoClient.DeleteItem(

```

```

            request);
        if (result.IsSuccess()) {
            std::cout << "\nRemoved \" " << title << "\" from the database."
                << std::endl;
        }
        else {
            std::cerr << "Failed to delete the movie: "
                << result.GetError().GetMessage()
                << std::endl;
        }
    }
}

return true;
}

//! Routine to convert a JsonView object to an attribute map.
/*!
\sa movieJsonViewToAttributeMap()
\param jsonView: Json view object.
\return map: Map that can be used in a DynamoDB request.
*/
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
AwsDoc::DynamoDB::movieJsonViewToAttributeMap(
    const Aws::Utils::Json::JsonView &jsonView) {
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> result;

    if (jsonView.KeyExists(YEAR_KEY)) {
        result[YEAR_KEY].SetN(jsonView.GetInteger(YEAR_KEY));
    }
    if (jsonView.KeyExists(TITLE_KEY)) {
        result[TITLE_KEY].SetS(jsonView.GetString(TITLE_KEY));
    }
    if (jsonView.KeyExists(INFO_KEY)) {
        Aws::Map<Aws::String, const
std::shared_ptr<Aws::DynamoDB::Model::AttributeValue>> infoMap;
        Aws::Utils::Json::JsonView infoView = jsonView.GetObject(INFO_KEY);
        if (infoView.KeyExists(RATING_KEY)) {
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue =
std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
            attributeValue->SetN(infoView.GetDouble(RATING_KEY));
            infoMap.emplace(std::make_pair(RATING_KEY, attributeValue));
        }
        if (infoView.KeyExists(PLOT_KEY)) {
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue =
std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
            attributeValue->SetS(infoView.GetString(PLOT_KEY));
            infoMap.emplace(std::make_pair(PLOT_KEY, attributeValue));
        }
        result[INFO_KEY].SetM(infoMap);
    }

    return result;
}

//! Create a DynamoDB table.
/*!
\sa createDynamoDBTable()
\param tableName: The DynamoDB table's name.
\param clientConfiguration: Aws client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createDynamoDBTable(const Aws::String &tableName,
                                             const Aws::Client::ClientConfiguration
                                             &clientConfiguration) {

```

```
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

bool movieTableAlreadyExisted = false;

{
    Aws::DynamoDB::Model::CreateTableRequest request;

    Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
    yearAttributeDefinition.SetAttributeName(YEAR_KEY);
    yearAttributeDefinition.SetAttributeType(
        Aws::DynamoDB::Model::ScalarAttributeType::N);
    request.AddAttributeDefinitions(yearAttributeDefinition);

    Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
    yearAttributeDefinition.SetAttributeName(TITLE_KEY);
    yearAttributeDefinition.SetAttributeType(
        Aws::DynamoDB::Model::ScalarAttributeType::S);
    request.AddAttributeDefinitions(titleAttributeDefinition);

    Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
    yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
        Aws::DynamoDB::Model::KeyType::HASH);
    request.AddKeySchema(yearKeySchema);

    Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
    yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
        Aws::DynamoDB::Model::KeyType::RANGE);
    request.AddKeySchema(titleKeySchema);

    Aws::DynamoDB::Model::ProvisionedThroughput throughput;
    throughput.WithReadCapacityUnits(
        PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
        PROVISIONED_THROUGHPUT_UNITS);
    request.SetProvisionedThroughput(throughput);
    request.SetTableName(MOVIE_TABLE_NAME);

    std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
    const Aws::DynamoDB::Model::CreateTableOutcome &result =
dynamoClient.CreateTable(
    request);
    if (!result.IsSuccess()) {
        if (result.GetError().GetErrorCode() ==
            Aws::DynamoDB::Errors::RESOURCE_IN_USE) {
            std::cout << "Table already exists." << std::endl;
            movieTableAlreadyExisted = true;
        }
        else {
            std::cerr << "Failed to create table: "
                << result.GetError().GetMessage();
            return false;
        }
    }
}

// Wait for table to become active.
if (!movieTableAlreadyExisted) {
    std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
        << "' to become active...." << std::endl;
    if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
clientConfiguration)) {
        return false;
    }
    std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
        << std::endl;
}
```

```

        return true;
    }

///! Delete a DynamoDB table.
/*!
\sa deleteDynamoTable()
\param tableName: The DynamoDB table's name.
\param clientConfiguration: Aws client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteDynamoTable(const Aws::String &tableName,
                                            const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
dynamoClient.DeleteTable(
        request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
            << result.GetResult().GetTableDescription().GetTableName()
            << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
            << std::endl;
    }
    return result.IsSuccess();
}

///! Query a newly created DynamoDB table until it is active.
/*!
\sa waitTableActive()
\param waitTableActive: The DynamoDB table's name.
\param clientConfiguration: Aws client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                         const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
            request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
    }
}

```

```
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
return false;
}
```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [.GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a struct and methods that call DynamoDB actions.

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified
// table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// TableExists determines whether a DynamoDB table exists.
func (basics TableBasics) TableExists() (bool, error) {
    exists := true
    _, err := basics.DynamoDbClient.DescribeTable(
        context.TODO(), &dynamodb.DescribeTableInput{TableName:
    aws.String(basics.TableName)},
    )
    if err != nil {
        var notFoundEx *types.ResourceNotFoundException
        if errors.As(err, &notFoundEx) {
            log.Printf("Table %v does not exist.\n", basics.TableName)
            err = nil
        } else {
    }}
```

```

        log.Printf("Couldn't determine existence of table %v. Here's why: %v\n",
        basics.TableName, err)
    }
    exists = false
}
return exists, err
}

// CreateMovieTable creates a DynamoDB table with a composite primary key defined
// as
// a string sort key named `title`, and a numeric partition key named `year`.
// This function uses NewTableExistsWaiter to wait for the table to be created by
// DynamoDB before it returns.
func (basics TableBasics) CreateMovieTable() (*types.TableDescription, error) {
    var tableDesc *types.TableDescription
    table, err := basics.DynamoDbClient.CreateTable(context.TODO(),
    &dynamodb.CreateTableInput{
        AttributeDefinitions: []types.AttributeDefinition{
            {AttributeName: aws.String("year"),
             AttributeType: types.ScalarAttributeTypeN,
        },
        {AttributeName: aws.String("title"),
         AttributeType: types.ScalarAttributeTypeS,
        },
        KeySchema: []types.KeySchemaElement{
            {AttributeName: aws.String("year"),
             KeyType: types.KeyTypeHash,
        },
        {AttributeName: aws.String("title"),
         KeyType: types.KeyTypeRange,
        },
        TableName: aws.String(basics.TableName),
        ProvisionedThroughput: &types.ProvisionedThroughput{
            ReadCapacityUnits: aws.Int64(10),
            WriteCapacityUnits: aws.Int64(10),
        },
    })
    if err != nil {
        log.Printf("Couldn't create table %v. Here's why: %v\n", basics.TableName, err)
    } else {
        waiter := dynamodb.NewTableExistsWaiter(basics.DynamoDbClient)
        err = waiter.Wait(context.TODO(), &dynamodb.DescribeTableInput{
            TableName: aws.String(basics.TableName)}, 5*time.Minute)
        if err != nil {
            log.Printf("Wait for table exists failed. Here's why: %v\n", err)
        }
        tableDesc = table.TableDescription
    }
    return tableDesc, err
}

// ListTables lists the DynamoDB table names for the current account.
func (basics TableBasics) ListTables() ([]string, error) {
    var tableNames []string
    tables, err := basics.DynamoDbClient.ListTables(
        context.TODO(), &dynamodb.ListTablesInput{})
    if err != nil {
        log.Printf("Couldn't list tables. Here's why: %v\n", err)
    } else {
        tableNames = tables.TableNames
    }
    return tableNames, err
}

```

```
}

// AddMovie adds a movie the DynamoDB table.
func (basics TableBasics) AddMovie(movie Movie) error {
    item, err := attributevalue.MarshalMap(movie)
    if err != nil {
        panic(err)
    }
    _, err = basics.DynamoDbClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
        TableName: aws.String(basics.TableName), Item: item,
    })
    if err != nil {
        log.Printf("Couldn't add item to table. Here's why: %v\n", err)
    }
    return err
}

// UpdateMovie updates the rating and plot of a movie that already exists in the
// DynamoDB table. This function uses the `expression` package to build the update
// expression.
func (basics TableBasics) UpdateMovie(movie Movie) (map[string]map[string]interface{}, error) {
    var err error
    var response *dynamodb.UpdateItemOutput
    var attributeMap map[string]map[string]interface{}
    update := expression.Set(expression.Name("info.rating"),
        expression.Value(movie.Info["rating"]))
    update.Set(expression.Name("info.plot"), expression.Value(movie.Info["plot"]))
    expr, err := expression.NewBuilder().WithUpdate(update).Build()
    if err != nil {
        log.Printf("Couldn't build expression for update. Here's why: %v\n", err)
    } else {
        response, err = basics.DynamoDbClient.UpdateItem(context.TODO(),
            &dynamodb.UpdateItemInput{
                TableName:           aws.String(basics.TableName),
                Key:                 movie.GetKey(),
                ExpressionAttributeNames: expr.Names(),
                ExpressionAttributeValues: expr.Values(),
                UpdateExpression:      expr.Update(),
                ReturnValue:          types.ReturnValueUpdatedNew,
            })
        if err != nil {
            log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
        } else {
            err = attributevalue.UnmarshalMap(response.Attributes, &attributeMap)
            if err != nil {
                log.Printf("Couldn't unmarshall update response. Here's why: %v\n", err)
            }
        }
    }
    return attributeMap, err
}

// AddMovieBatch adds a slice of movies to the DynamoDB table. The function sends
// batches of 25 movies to DynamoDB until all movies are added or it reaches the
// specified maximum.
func (basics TableBasics) AddMovieBatch(movies []Movie, maxMovies int) (int, error) {
{
    var err error
    var item map[string]types.AttributeValue
```

```

written := 0
batchSize := 25 // DynamoDB allows a maximum batch size of 25 items.
start := 0
end := start + batchSize
for start < maxMovies && start < len(movies) {
    var writeReqs []types.WriteRequest
    if end > len(movies) {
        end = len(movies)
    }
    for _, movie := range movies[start:end] {
        item, err = attributevalue.MarshalMap(movie)
        if err != nil {
            log.Printf("Couldn't marshal movie %v for batch writing. Here's why: %v\n",
                      movie.Title, err)
        } else {
            writeReqs = append(
                writeReqs,
                types.WriteRequest{PutRequest: &types.PutRequest{Item: item}},
            )
        }
    }
    _, err = basics.DynamoDbClient.BatchWriteItem(context.TODO(),
&dynamodb.BatchWriteItemInput{
    RequestItems: map[string][]types.WriteRequest{basics.TableName: writeReqs}})
    if err != nil {
        log.Printf("Couldn't add a batch of movies to %v. Here's why: %v\n",
                  basics.TableName, err)
    } else {
        written += len(writeReqs)
    }
    start = end
    end += batchSize
}

return written, err
}

// GetMovie gets movie data from the DynamoDB table by using the primary composite
// key
// made of title and year.
func (basics TableBasics) GetMovie(title string, year int) (Movie, error) {
    movie := Movie{Title: title, Year: year}
    response, err := basics.DynamoDbClient.GetItem(context.TODO(),
&dynamodb.GetItemInput{
    Key: movie.GetKey(), TableName: aws.String(basics.TableName),
})
    if err != nil {
        log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
    } else {
        err = attributevalue.UnmarshalMap(response.Item, &movie)
        if err != nil {
            log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
        }
    }
    return movie, err
}

// Query gets all movies in the DynamoDB table that were released in the specified
// year.
// The function uses the `expression` package to build the key condition expression
// that is used in the query.
func (basics TableBasics) Query(releaseYear int) ([]Movie, error) {

```

```

var err error
var response *dynamodb.QueryOutput
var movies []Movie
keyEx := expression.Key("year").Equal(expression.Value(releaseYear))
expr, err := expression.NewBuilder().WithKeyCondition(keyEx).Build()
if err != nil {
    log.Printf("Couldn't build expression for query. Here's why: %v\n", err)
} else {
    response, err = basics.DynamoDbClient.Query(context.TODO(), &dynamodb.QueryInput{
        TableName: aws.String(basics.TableName),
        ExpressionAttributeNames: expr.Names(),
        ExpressionAttributeValues: expr.Values(),
        KeyConditionExpression: expr.KeyCondition(),
    })
    if err != nil {
        log.Printf("Couldn't query for movies released in %v. Here's why: %v\n",
            releaseYear, err)
    } else {
        err = attributevalue.UnmarshalListOfMaps(response.Items, &movies)
        if err != nil {
            log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
        }
    }
}
return movies, err
}

// Scan gets all movies in the DynamoDB table that were released in a range of
// years
// and projects them to return a reduced set of fields.
// The function uses the `expression` package to build the filter and projection
// expressions.
func (basics TableBasics) Scan(startYear int, endYear int) ([]Movie, error) {
    var movies []Movie
    var err error
    var response *dynamodb.ScanOutput
    filtEx := expression.Name("year").Between(expression.Value(startYear),
        expression.Value(endYear))
    projEx := expression.NamesList(
        expression.Name("year"), expression.Name("title"),
        expression.Name("info.rating"))
    expr, err :=
        expression.NewBuilder().WithFilter(filtEx).WithProjection(projEx).Build()
    if err != nil {
        log.Printf("Couldn't build expressions for scan. Here's why: %v\n", err)
    } else {
        response, err = basics.DynamoDbClient.Scan(context.TODO(), &dynamodb.ScanInput{
            TableName: aws.String(basics.TableName),
            ExpressionAttributeNames: expr.Names(),
            ExpressionAttributeValues: expr.Values(),
            FilterExpression: expr.Filter(),
            ProjectionExpression: expr.Projection(),
        })
        if err != nil {
            log.Printf("Couldn't scan for movies released between %v and %v. Here's why: %v
\n",
                startYear, endYear, err)
        } else {
            err = attributevalue.UnmarshalListOfMaps(response.Items, &movies)
            if err != nil {
                log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
            }
        }
    }
}

```

```
    return movies, err
}

// DeleteMovie removes a movie from the DynamoDB table.
func (basics TableBasics) DeleteMovie(movie Movie) error {
    _, err := basics.DynamoDbClient.DeleteItem(context.TODO(),
&dynamodb.DeleteItemInput{
    TableName: aws.String(basics.TableName), Key: movie.GetKey(),
})
if err != nil {
    log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
err)
}
return err
}

// DeleteTable deletes the DynamoDB table and all of its data.
func (basics TableBasics) DeleteTable() error {
    _, err := basics.DynamoDbClient.DeleteTable(context.TODO(),
&dynamodb.DeleteTableInput{
    TableName: aws.String(basics.TableName)})
if err != nil {
    log.Printf("Couldn't delete table %v. Here's why: %v\n", basics.TableName, err)
}
return err
}
```

Run an interactive scenario to create the table and perform actions on it.

```
// RunMovieScenario is an interactive example that shows you how to use the AWS SDK
for Go
// to create and use an Amazon DynamoDB table that stores data about movies.
//
// 1. Create a table that can hold movie data.
// 2. Put, get, and update a single movie in the table.
// 3. Write movie data to the table from a sample JSON file.
// 4. Query for movies that were released in a given year.
// 5. Scan for movies that were released in a range of years.
// 6. Delete a movie from the table.
// 7. Delete the table.
//
// This example creates a DynamoDB service client from the specified sdkConfig so
// that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// The specified movie sampler is used to get sample data from a URL that is loaded
// into the named table.
func RunMovieScenario(
    sdkConfig aws.Config, questioner demotools.IQuestioner, tableName string,
    movieSampler actions.IMovieSampler) {
    defer func() {
        if r := recover(); r != nil {
            fmt.Printf("Something went wrong with the demo.%s")
    }
}
```

```
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon DynamoDB getting started demo.")
log.Println(strings.Repeat("-", 88))

tableBasics := actions.TableBasics{TableName: tableName,
    DynamoDbClient: dynamodb.NewFromConfig(sdkConfig)}

exists, err := tableBasics.TableExists()
if err != nil {
    panic(err)
}
if !exists {
    log.Printf("Creating table %v...\n", tableName)
    _, err = tableBasics.CreateMovieTable()
    if err != nil {
        panic(err)
    } else {
        log.Printf("Created table %v.\n", tableName)
    }
} else {
    log.Printf("Table %v already exists.\n", tableName)
}

var customMovie actions.Movie
customMovie.Title = questioner.Ask("Enter a movie title to add to the table:",
    []demotools.IAnswerValidator{demotools.NotEmpty{}})
customMovie.Year = questioner.AskInt("What year was it released?",
    []demotools.IAnswerValidator{demotoools.NotEmpty{}, demotools.InIntRange{
        Lower: 1900, Upper: 2030}})
customMovie.Info = map[string]interface{}{}
customMovie.Info["rating"] = questioner.AskFloat64(
    "Enter a rating between 1 and 10:", []demotools.IAnswerValidator{
        demotools.NotEmpty{}, demotools.InFloatRange{Lower: 1, Upper: 10}})
customMovie.Info["plot"] = questioner.Ask("What's the plot? ",
    []demotools.IAnswerValidator{demotools.NotEmpty{}})
err = tableBasics.AddMovie(customMovie)
if err == nil {
    log.Printf("Added %v to the movie table.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's update your movie. You previously rated it %v.\n",
    customMovie.Info["rating"])
customMovie.Info["rating"] = questioner.AskFloat64(
    "What new rating would you give it?", []demotools.IAnswerValidator{
        demotools.NotEmpty{}, demotools.InFloatRange{Lower: 1, Upper: 10}})
log.Printf("You summarized the plot as '%v'.\n", customMovie.Info["plot"])
customMovie.Info["plot"] = questioner.Ask("What would you say now?",
    []demotools.IAnswerValidator{demotools.NotEmpty{}})
attributes, err := tableBasics.UpdateMovie(customMovie)
if err == nil {
    log.Printf("Updated %v with new values.\n", customMovie.Title)
    for _, attVal := range attributes {
        for valKey, val := range attVal {
            log.Printf("\t%v: %v\n", valKey, val)
        }
    }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Getting movie data from %v and adding 250 movies to the table....\n",
    movieSampler.GetURL())
movies := movieSampler.GetSampleMovies()
written, err := tableBasics.AddMovieBatch(movies, 250)
```

```
if err != nil {
    panic(err)
} else {
    log.Printf("Added %v movies to the table.\n", written)
}

show := 10
if show > written {
    show = written
}
log.Printf("The first %v movies in the table are:", show)
for index, movie := range movies[:show] {
    log.Printf("\t%v. %v\n", index+1, movie.Title)
}
movieIndex := questioner.AskInt(
    "Enter the number of a movie to get info about it: ",
    []demotools.IAnswerValidator{
        demotools.InIntRange{Lower: 1, Upper: show}},
)
movie, err := tableBasics.GetMovie(movies[movieIndex-1].Title,
movies[movieIndex-1].Year)
if err == nil {
    log.Println(movie)
}
log.Println(strings.Repeat("-", 88))

log.Println("Let's get a list of movies released in a given year.")
releaseYear := questioner.AskInt("Enter a year between 1972 and 2018: ",
    []demotools.IAnswerValidator{demotoools.InIntRange{Lower: 1972, Upper: 2018}},
)
releases, err := tableBasics.Query(releaseYear)
if err == nil {
    if len(releases) == 0 {
        log.Printf("I couldn't find any movies released in %v!\n", releaseYear)
    } else {
        for _, movie = range releases {
            log.Println(movie)
        }
    }
}
log.Println(strings.Repeat("-", 88))

log.Println("Now let's scan for movies released in a range of years.")
startYear := questioner.AskInt("Enter a year: ", []demotools.IAnswerValidator{
    demotools.InIntRange{Lower: 1972, Upper: 2018}})
endYear := questioner.AskInt("Enter another year: ", []demotools.IAnswerValidator{
    demotools.InIntRange{Lower: 1972, Upper: 2018}})
releases, err = tableBasics.Scan(startYear, endYear)
if err == nil {
    if len(releases) == 0 {
        log.Printf("I couldn't find any movies released between %v and %v!\n",
startYear, endYear)
    } else {
        log.Printf("Found %v movies. In this list, the plot is <nil> because "+
            "we used a projection expression when scanning for items to return only "+
            "the title, year, and rating.\n", len(releases))
        for _, movie = range releases {
            log.Println(movie)
        }
    }
}
log.Println(strings.Repeat("-", 88))

var tables []string
if questioner.AskBool("Do you want to list all of your tables? (y/n) ", "y") {
    tables, err = tableBasics.ListTables()
```

```
if err == nil {
    log.Printf("Found %v tables:", len(tables))
    for _, table := range tables {
        log.Printf("\t%v", table)
    }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's remove your movie '%v'.\n", customMovie.Title)
if questioner.AskBool("Do you want to delete it from the table? (y/n) ", "y") {
    err = tableBasics.DeleteMovie(customMovie)
}
if err == nil {
    log.Printf("Deleted %v.\n", customMovie.Title)
}

if questioner.AskBool("Delete the table, too? (y/n)", "y") {
    err = tableBasics.DeleteTable()
} else {
    log.Println("Don't forget to delete the table when you're done or you might " +
        "incur charges on your account.")
}
if err == nil {
    log.Printf("Deleted table %v.\n", tableBasics.TableName)
}

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
```

- For API details, see the following topics in *AWS SDK for Go API Reference*.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a DynamoDB table.

```
// Create a table with a Sort key.
public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
```

```

ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

// Define attributes.
attributeDefinitions.add(AttributeDefinition.builder()
    .attributeName("year")
    .attributeType("N")
    .build());

attributeDefinitions.add(AttributeDefinition.builder()
    .attributeName("title")
    .attributeType("S")
    .build());

ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
KeySchemaElement key = KeySchemaElement.builder()
    .attributeName("year")
    .keyType(KeyType.HASH)
    .build();

KeySchemaElement key2 = KeySchemaElement.builder()
    .attributeName("title")
    .keyType(KeyType.RANGE)
    .build();

// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(new Long(10))
        .writeCapacityUnits(new Long(10))
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
    dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " +newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
}

```

Create a helper function to download and extract the sample JSON file.

```

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)

```

```
.build();

DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
JsonParser parser = new JsonFactory().createParser(new File(fileName));
com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
Iterator<JsonNode> iter = rootNode.iterator();
ObjectNode currentNode;
int t = 0 ;
while (iter.hasNext()) {
    // Only add 200 Movies to the table.
    if (t == 200)
        break ;
    currentNode = (ObjectNode) iter.next();

    int year = currentNode.path("year").asInt();
    String title = currentNode.path("title").asText();
    String info = currentNode.path("info").toString();

    Movies movies = new Movies();
    movies.setYear(year);
    movies.setTitle(title);
    movies.setInfo(info);

    // Put the data into the Amazon DynamoDB Movie table.
    mappedTable.putItem(movies);
    t++;
}
}
```

Get an item from a table.

```
public static void getItem(DynamoDbClient ddb) {

    HashMap<String,AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put("year", AttributeValue.builder()
        .n("1933")
        .build());

    keyToGet.put("title", AttributeValue.builder()
        .s("King Kong")
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName("Movies")
        .build();

    try {
        Map<String,AttributeValue> returnedItem = ddb.getItem(request).item();

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
                    returnedItem.get(key1).toString());
            }
        } else {
            System.out.format("No item found with the key %s!\n", "year");
        }
    }
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

Full example.

```
/***
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Creates the Amazon DynamoDB Movie table with partition and sort key.
 * 2. Puts data into the Amazon DynamoDB table from a JSON document using the Enhanced client.
 * 3. Gets data from the Movie table.
 * 4. Adds a new item.
 * 5. Updates an item.
 * 6. Uses a Scan to query items using the Enhanced client.
 * 7. Queries all items where the year is 2013 using the Enhanced Client.
 * 8. Deletes the table.
 */

public class Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");
    public static void main(String[] args) throws IOException {
        final String usage = "\n" +
            "Usage:\n" +
            "      <fileName>\n\n" +
            "Where:\n" +
            "      fileName - The path to the moviedata.json file that you can download from the Amazon DynamoDB Developer Guide.\n" ;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = "Movies";
        String fileName = args[0];
        ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .credentialsProvider(credentialsProvider)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon DynamoDB example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Creating an Amazon DynamoDB table named Movies with a key named year and a sort key named title.");
        createTable(ddb, tableName);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Loading data into the Amazon DynamoDB table.");
loadData(ddb, tableName, fileName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Getting data from the Movie table.");
getItem(ddb) ;
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Putting a record into the Amazon DynamoDB table.");
putRecord(ddb);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Updating a record.");
updateTableItem(ddb, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Scanning the Amazon DynamoDB table.");
scanMovies(ddb, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Querying the Movies released in 2013.");
queryTable(ddb);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Deleting the Amazon DynamoDB table.");
deleteDynamoDBTable(ddb, tableName);
System.out.println(DASHES);

ddb.close();
}

// Create a table with a Sort key.
public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
    KeySchemaElement key = KeySchemaElement.builder()
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE)
        .build();
}
```

```
// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(new Long(10))
        .writeCapacityUnits(new Long(10))
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
    dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " +newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Query the table.
public static void queryTable(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
    .dynamoDbClient(ddb)
    .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        QueryConditional queryConditional = QueryConditional
            .keyEqualTo(Key.builder()
            .partitionValue(2013)
            .build());

        // Get items in the table and write out the ID value.
        Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
        String result="";

        while (results.hasNext()) {
            Movies rec = results.next();
            System.out.println("The title of the movie is "+rec.getTitle());
            System.out.println("The movie information is "+rec.getInfo());
        }
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```

// Scan the table.
public static void scanMovies(DynamoDbClient ddb, String tableName) {
    System.out.println("***** Scanning all movies.\n");
    try{
        DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        Iterator<Movies> results = custTable.scan().items().iterator();
        while (results.hasNext()) {
            Movies rec = results.next();
            System.out.println("The movie title is "+rec.getTitle());
            System.out.println("The movie year is " +rec.getYear());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

    DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0 ;
    while (iter.hasNext()) {
        // Only add 200 Movies to the table.
        if (t == 200)
            break ;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        Movies movies = new Movies();
        movies.setYear(year);
        movies.setTitle(title);
        movies.setInfo(info);

        // Put the data into the Amazon DynamoDB Movie table.
        mappedTable.putItem(movies);
        t++;
    }
}

// Update the record to include show only directors.
public static void updateTableItem(DynamoDbClient ddb, String tableName){
    HashMap<String,AttributeValue> itemKey = new HashMap<>();
    itemKey.put("year", AttributeValue.builder().n("1933").build());
    itemKey.put("title", AttributeValue.builder().s("King Kong").build());
}

```

```
    HashMap<String,AttributeValueUpdate> updatedValues = new HashMap<>();
    updatedValues.put("info", AttributeValueUpdate.builder()
        .value(AttributeValue.builder().s("{\"directors\":[\"Merian C. Cooper
\"],\"Ernest B. Schoedsack\"]}").build())
        .action(AttributeAction.PUT)
        .build());

    UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(itemKey)
        .attributeUpdates(updatedValues)
        .build();

    try {
        ddb.updateItem(request);
    } catch (ResourceNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println("Item was updated!");
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName +" was successfully deleted!");
}

public static void putRecord(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

        DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));

        // Populate the Table.
        Movies record = new Movies();
        record.setYear(2020);
        record.setTitle("My Movie2");
        record.setInfo("no info");
        table.putItem(record);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Added a new movie to the table.");
}

public static void getItem(DynamoDbClient ddb) {
```

```
HashMap<String,AttributeValue> keyToGet = new HashMap<>();
keyToGet.put("year", AttributeValue.builder()
    .n("1933")
    .build());

keyToGet.put("title", AttributeValue.builder()
    .s("King Kong")
    .build());

GetItemRequest request = GetItemRequest.builder()
    .key(keyToGet)
    .tableName("Movies")
    .build();

try {
    Map<String,AttributeValue> returnedItem = ddb.getItem(request).item();

    if (returnedItem != null) {
        Set<String> keys = returnedItem.keySet();
        System.out.println("Amazon DynamoDB table attributes: \n");

        for (String key1 : keys) {
            System.out.format("%s: %s\n", key1,
                returnedItem.get(key1).toString());
        }
    } else {
        System.out.format("No item found with the key %s!\n", "year");
    }

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a DynamoDB client.

```
// Create the DynamoDB service client module using ES6 syntax.  
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";  
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";  
// Create an Amazon DynamoDB service client object.  
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create a DynamoDB document client.

```
// Create a service client module using ES6 syntax.  
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";  
import { ddbClient } from "./ddbClient.js";  
  
const marshallOptions = {  
    // Whether to automatically convert empty strings, blobs, and sets to `null`.  
    convertEmptyValues: false, // false, by default.  
    // Whether to remove undefined values while marshalling.  
    removeUndefinedValues: true, // false, by default.  
    // Whether to convert typeof object to map attribute.  
    convertClassInstanceToMap: false, // false, by default.  
};  
  
const unmarshallOptions = {  
    // Whether to return numbers as a string instead of converting them to native  
    // JavaScript numbers.  
    wrapNumbers: false, // false, by default.  
};  
  
// Create the DynamoDB document client.  
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {  
    marshallOptions,  
    unmarshallOptions,  
});  
  
export { ddbDocClient };
```

Run the scenario.

```
import fs from "fs";  
import { splitEvery } from "ramda";  
import {  
    PutCommand,  
    GetCommand,  
    UpdateCommand,  
    BatchWriteCommand,  
    DeleteCommand,  
    ScanCommand,  
    QueryCommand,  
} from "@aws-sdk/lib-dynamodb";  
import {  
    CreateTableCommand,  
    DeleteTableCommand,  
    waitUntilTableExists,  
    waitUntilTableNotExists,  
} from "@aws-sdk/client-dynamodb";  
  
import { ddbClient } from "../libs/ddbClient.js";  
import { ddbDocClient } from "../libs/ddbDocClient.js";  
  
/**  
 * @param {string} tableName
```

```
/*
const createTable = async (tableName) => {
    await ddbClient.send(
        new CreateTableCommand({
            AttributeDefinitions: [
                {
                    AttributeName: "year",
                    AttributeType: "N",
                },
                {
                    AttributeName: "title",
                    AttributeType: "S",
                },
            ],
            KeySchema: [
                {
                    AttributeName: "year",
                    KeyType: "HASH",
                },
                {
                    AttributeName: "title",
                    KeyType: "RANGE",
                },
            ],
            // Enables "on-demand capacity mode".
            BillingMode: "PAY_PER_REQUEST",
            TableName: tableName,
        })
    );
    await waitUntilTableExists(
        { client: ddbClient, maxWaitTime: 15, maxDelay: 2, minDelay: 1 },
        { TableName: tableName }
    );
};

/** 
 * @param {string} tableName
 * @param {Record<string, any> | undefined} attributes
 */
const putItem = async (tableName, attributes) => {
    const command = new PutCommand({
        TableName: tableName,
        Item: attributes,
    });

    await ddbDocClient.send(command);
};

/** 
 * @param {string} tableName
 * @param {string} filePath
 * @returns { { movieCount: number } } The number of movies written to the database.
 */
const batchWriteMoviesFromFile = async (tableName, filePath) => {
    const fileContents = fs.readFileSync(filePath);
    const movies = JSON.parse(fileContents, "utf8");

    // Map movies to RequestItems.
    const putMovieRequestItems = movies.map(({ year, title, info }) => ({
        PutRequest: { Item: { year, title, info } },
    }));
}
```

```
// Organize RequestItems into batches of 25. 25 is the max number of items in a
batch request.
const putMovieBatches = splitEvery(25, putMovieRequestItems);
const batchCount = putMovieBatches.length;

// Map batches to promises.
const batchRequests = putMovieBatches.map(async (batch, i) => {
    const command = new BatchWriteCommand({
        RequestItems: [
            [tableName]: batch,
        ],
    });

    await ddbDocClient.send(command).then(() => {
        console.log(
            `Wrote batch ${i + 1} of ${batchCount} with ${batch.length} items.`
        );
    });
});

// Wait for all batch requests to resolve.
await Promise.all(batchRequests);

return { movieCount: movies.length };
};

/***
 *
 * @param {string} tableName
 * @param {{
 * existingMovieName: string,
 * existingMovieYear: string,
 * newMoviePlot: string,
 * newMovieRank: string}} keyUpdate
 */
const updateMovie = async (
    tableName,
    { existingMovieName, existingMovieYear, newMoviePlot, newMovieRank }
) => {
    await ddbClient.send(
        new UpdateCommand({
            TableName: tableName,
            Key: {
                title: existingMovieName,
                year: existingMovieYear,
            },
            // Define expressions for the new or updated attributes.
            ExpressionAttributeNames: { "#r": "rank" },
            UpdateExpression: "set info.plot = :p, info.#r = :r",
            ExpressionAttributeValues: {
                ":p": newMoviePlot,
                ":r": newMovieRank,
            },
            ReturnValues: "ALL_NEW",
        })
    );
};

/***
 * @param {{ title: string, info: { plot: string, rank: number }, year: number }} movie
 */
const logMovie = (movie) => {
    console.log(` | Title: "${movie.title}"`);
    console.log(` | Plot: "${movie.info.plot}"`);
    console.log(` | Year: ${movie.year}`);
};
```

```
        console.log(` | Rank: ${movie.info.rank}`);
    };

/** 
 * @param {{ title: string, info: { plot: string, rank: number }, year: number }[]} movies
 */
const logMovies = (movies) => {
    console.log("\n");
    movies.forEach((movie, i) => {
        if (i > 0) {
            console.log("-".repeat(80));
        }

        logMovie(movie);
    });
};

/** 
 * @param {string} tableName
 * @param {string} title
 * @param {number} year
 * @returns
 */
const getMovie = async (tableName, title, year) => {
    const { Item } = await ddbDocClient.send(
        new GetCommand({
            TableName: tableName,
            Key: {
                title,
                year,
            },
            // By default, reads are eventually consistent. "ConsistentRead: true"
            // represents
            // a strongly consistent read. This guarantees that the most up-to-date data
            // is returned. It
            // can also result in higher latency and a potential for server errors.
            ConsistentRead: true,
        })
    );

    return Item;
};

/** 
 * @param {string} tableName
 * @param {{ title: string, year: number }} key
 */
const deleteMovie = async (tableName, key) => {
    await ddbDocClient.send(
        new DeleteCommand({
            TableName: tableName,
            Key: key,
        })
    );
};

/** 
 * @param {string} tableName
 * @param {number} startYear
 * @param {number} endYear
 * @param {Record<string, any>} startKey

```

```
* @returns {Promise<{}[]>}
*/
const findMoviesBetweenYears = async (
  tableName,
  startYear,
  endYear,
  startKey = undefined
) => {
  const { Items, LastEvaluatedKey } = await ddbClient.send(
    new ScanCommand({
      ConsistentRead: true,
      TableName: tableName,
      ExpressionAttributeNames: { "#y": "year" },
      FilterExpression: "#y BETWEEN :y1 AND :y2",
      ExpressionAttributeValues: { ":y1": startYear, ":y2": endYear },
      ExclusiveStartKey: startKey,
    })
  );
  if (LastEvaluatedKey) {
    return Items.concat(
      await findMoviesBetweenYears(
        tableName,
        startYear,
        endYear,
        LastEvaluatedKey
      )
    );
  } else {
    return Items;
  }
};

/***
 *
 * @param {string} tableName
 * @param {number} year
 * @returns
 */
const queryMoviesByYear = async (tableName, year) => {
  const command = new QueryCommand({
    ConsistentRead: true,
    ExpressionAttributeNames: { "#y": "year" },
    TableName: tableName,
    ExpressionAttributeValues: {
      ":y": year,
    },
    KeyConditionExpression: "#y = :y",
  });
  const { Items } = await ddbDocClient.send(command);
  return Items;
};

/***
 *
 * @param {*} tableName
 */
const deleteTable = async (tableName) => {
  await ddbDocClient.send(new DeleteTableCommand({ TableName: tableName }));
  await waitUntilTableNotExists(
    {
      client: ddbClient,
      maxWaitTime: 10,
      maxDelay: 2,
```

```
        minDelay: 1,
    },
    { TableName: tableName }
);
};

export const runScenario = async ({
    tableName,
    newMovieName,
    newMovieYear,
    existingMovieName,
    existingMovieYear,
    newMovieRank,
    newMoviePlot,
    moviesPath,
}) => {
    console.log(`Creating table named: ${tableName}`);
    await createTable(tableName);
    console.log(`\nTable created.`);

    console.log(`\nAdding "${newMovieName}" to ${tableName}.`);
    await putItem(tableName, { title: newMovieName, year: newMovieYear });
    console.log("\nSuccess - single movie added.");

    console.log("\nWriting hundreds of movies in batches.");
    const { movieCount } = await batchWriteMoviesFromFile(tableName, moviesPath);
    console.log(`\nWrote ${movieCount} movies to database.`);

    console.log(`\nGetting "${existingMovieName}"`);
    const originalMovie = await getMovie(
        tableName,
        existingMovieName,
        existingMovieYear
    );
    logMovie(originalMovie);

    console.log(`\nUpdating "${existingMovieName}" with a new plot and rank.`);
    await updateMovie(tableName, {
        existingMovieName,
        existingMovieYear,
        newMoviePlot,
        newMovieRank,
    });
    console.log(`\n"${existingMovieName}" updated.`);

    console.log(`\nGetting latest info for "${existingMovieName}"`);
    const updatedMovie = await getMovie(
        tableName,
        existingMovieName,
        existingMovieYear
    );
    logMovie(updatedMovie);

    console.log(`\nDeleting "${newMovieName}.");
    await deleteMovie(tableName, { title: newMovieName, year: newMovieYear });
    console.log(`\n"${newMovieName} deleted.`);

    const [scanY1, scanY2] = [1985, 2003];
    console.log(
        `\nScanning ${tableName} for movies that premiered between ${scanY1} and
${scanY2}.`);
    const scannedMovies = await findMoviesBetweenYears(tableName, scanY1, scanY1);
    logMovies(scannedMovies);

    const queryY = 2003;
```

```
console.log(`Querying ${tableName} for movies that premiered in ${queryY}.`);  
const queriedMovies = await queryMoviesByYear(tableName, queryY);  
logMovies(queriedMovies);  
  
console.log(`Deleting ${tableName}.`);  
await deleteTable(tableName);  
console.log(`$${tableName} deleted.`);  
};  
  
const main = async () => {  
  const args = {  
    tableName: "myNewTable",  
    newMovieName: "myMovieName",  
    newMovieYear: 2022,  
    existingMovieName: "This Is the End",  
    existingMovieYear: 2013,  
    newMovieRank: 200,  
    newMoviePlot: "A coder cracks code...",  
    moviesPath: "../../../../resources/sample_files/movies.json",  
  };  
  
  try {  
    await runScenario(args);  
  } catch (err) {  
    // Some extra error handling here to be sure the table is cleaned up if  
    // something  
    // goes wrong during the scenario run.  
  
    console.error(err);  
  
    const tableName = args.tableName;  
  
    if (tableName) {  
      console.log(`Attempting to delete ${tableName}`);  
      await ddbClient  
        .send(new DeleteTableCommand({ TableName: tableName }))  
        .then(() => console.log(`\n${tableName} deleted.`))  
        .catch((err) => console.error(`\nFailed to delete ${tableName}.`, err));  
    }  
  }  
};  
  
export { main };
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a DynamoDB table.

```
suspend fun createScenarioTable(tableNameVal: String, key: String) {

    val attDef = AttributeDefinition {
        attributeName = key
        attributeType = ScalarAttributeType.N
    }

    val attDef1 = AttributeDefinition {
        attributeName = "title"
        attributeType = ScalarAttributeType.S
    }

    val keySchemaVal = KeySchemaElement {
        attributeName = key
        keyType = KeyType.Hash
    }

    val keySchemaVal1 = KeySchemaElement {
        attributeName = "title"
        keyType = KeyType.Range
    }

    val provisionedVal = ProvisionedThroughput {
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }

    val request = CreateTableRequest {
        attributeDefinitions = listOf(attDef, attDef1)
        keySchema = listOf(keySchemaVal, keySchemaVal1)
        provisionedThroughput = provisionedVal
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->

        val response = ddb.createTable(request)
        ddb.waitUntilTableExists { // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
${response.tableDescription?.tableArn}")
    }
}
```

Create a helper function to download and extract the sample JSON file.

```
// Load data into the table.
suspend fun loadData(tableName: String, fileName: String) {
```

```
val parser = JsonFactory().createParser(File(fileName))
val rootNode = ObjectMapper().readTree<JsonNode>(parser)
val iter: Iterator<JsonNode> = rootNode.iterator()
var currentNode: ObjectNode

var t = 0
while (iter.hasNext()) {

    if (t == 50)
        break

    currentNode = iter.next() as ObjectNode
    val year = currentNode.path("year").asInt()
    val title = currentNode.path("title").asText()
    val info = currentNode.path("info").toString()
    putMovie(tableName, year, title, info)
    t++
}
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request = PutItemRequest {
        tableName = tableNameVal
        item = itemValues
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added '$title' to the Movie table.")
    }
}
```

Get an item from a table.

```
suspend fun getMovie(tableNameVal: String, keyName: String, keyVal: String) {

    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request = GetItemRequest {
        key = keyToGet
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

```
        }
    }
}
```

Full example.

```
suspend fun main(args: Array<String>) {

    val usage = """
        Usage:
            <fileName>

        Where:
            fileName - The path to the moviedata.json you can download from the
            Amazon DynamoDB Developer Guide.
        """

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    // Get the moviedata.json from the Amazon DynamoDB Developer Guide.
    val tableName = "Movies"
    val fileName = args[0]
    val partitionAlias = "#a"

    println("Creating an Amazon DynamoDB table named Movies with a key named id and
a sort key named title.")
    createScenarioTable(tableName, "year")
    loadData(tableName, fileName)
    getMovie(tableName, "year", "1933")
    scanMovies(tableName)
    val count = queryMovieTable(tableName, "year", partitionAlias)
    println("There are $count Movies released in 2013.")
    deleteIssuesTable(tableName)
}

suspend fun createScenarioTable(tableNameVal: String, key: String) {

    val attDef = AttributeDefinition {
        attributeName = key
        attributeType = ScalarAttributeType.N
    }

    val attDef1 = AttributeDefinition {
        attributeName = "title"
        attributeType = ScalarAttributeType.S
    }

    val keySchemaVal = KeySchemaElement {
        attributeName = key
        keyType = KeyType.Hash
    }

    val keySchemaVal1 = KeySchemaElement {
        attributeName = "title"
        keyType = KeyType.Range
    }

    val provisionedVal = ProvisionedThroughput {
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }
}
```

```
val request = CreateTableRequest {
    attributeDefinitions = listOf(attDef, attDef1)
    keySchema = listOf(keySchemaVal, keySchemaVal1)
    provisionedThroughput = provisionedVal
    tableName = tableNameVal
}

DynamoDbClient { region = "us-east-1" }.use { ddb ->

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists { // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
    ${response.tableDescription?.tableArn}")
}
}

// Load data into the table.
suspend fun loadData(tableName: String, fileName: String) {

    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {

        if (t == 50)
            break

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request = PutItemRequest {
        tableName = tableNameVal
        item = itemValues
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}
```

```
suspend fun getMovie(tableNameVal: String, keyName: String, keyVal: String) {  
  
    val keyToGet = mutableMapOf<String, AttributeValue>()  
    keyToGet[keyName] = AttributeValue.N(keyVal)  
    keyToGet["title"] = AttributeValue.S("King Kong")  
  
    val request = GetItemRequest {  
        key = keyToGet  
        tableName = tableNameVal  
    }  
  
    DynamoDbClient { region = "us-east-1" }.use { ddb ->  
        val returnedItem = ddb.getItem(request)  
        val numbersMap = returnedItem.item  
        numbersMap?.forEach { key1 ->  
            println(key1.key)  
            println(key1.value)  
        }  
    }  
}  
  
suspend fun deleteIssuesTable(tableNameVal: String) {  
  
    val request = DeleteTableRequest {  
        tableName = tableNameVal  
    }  
  
    DynamoDbClient { region = "us-east-1" }.use { ddb ->  
        ddb.deleteTable(request)  
        println("$tableNameVal was deleted")  
    }  
}  
  
suspend fun queryMovieTable(  
    tableNameVal: String,  
    partitionKeyName: String,  
    partitionAlias: String  
): Int {  
  
    val attrNameAlias = mutableMapOf<String, String>()  
    attrNameAlias[partitionAlias] = "year"  
  
    // Set up mapping of the partition name with the value.  
    val attrValues = mutableMapOf<String, AttributeValue>()  
    attrValues[":$partitionKeyName"] = AttributeValue.N("2013")  
  
    val request = QueryRequest {  
        tableName = tableNameVal  
        keyConditionExpression = "$partitionAlias = :$partitionKeyName"  
        expressionAttributeNames = attrNameAlias  
        this.expressionAttributeValues = attrValues  
    }  
  
    DynamoDbClient { region = "us-east-1" }.use { ddb ->  
        val response = ddb.query(request)  
        return response.count  
    }  
}  
  
suspend fun scanMovies(tableNameVal: String) {  
  
    val request = ScanRequest {  
        tableName = tableNameVal  
    }  
  
    DynamoDbClient { region = "us-east-1" }.use { ddb ->
```

```
    val response = ddb.scan(request)
    response.items?.forEach { item ->
        item.keys.forEach { key ->
            println("The key name is $key\n")
            println("The value is ${item[key]}")
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace DynamoDb\Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;
use DynamoDb\DynamoDBService;

class GettingStartedWithDynamoDB
{
    public function run()
    {
        echo("-----\n");
        print("Welcome to the Amazon DynamoDB getting started demo using PHP!\n");
        echo("-----\n");

        $uuid = uniqid();
        $service = new DynamoDBService();

        $tableName = "ddb_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );
        echo "Waiting for table...";
```

```
$service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
$tableName]);
echo "table $tableName found!\n";

echo "What's the name of the last movie you watched?\n";
while (empty($movieName)) {
    $movieName = testable_readline("Movie name: ");
}
echo "And what year was it released?\n";
$movieYear = "year";
while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
    $movieYear = testable_readline("Year released: ");
}

 getService->putItem([
    'Item' => [
        'year' => [
            'N' => "$movieYear",
        ],
        'title' => [
            'S' => $movieName,
        ],
    ],
    'TableName' => $tableName,
]);

echo "How would you rate the movie from 1-10?\n";
$rating = 0;
while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 || $rating > 10) {
    $rating = testable_readline("Rating (1-10): ");
}
echo "What was the movie about?\n";
while (empty($plot)) {
    $plot = testable_readline("Plot summary: ");
}
$key = [
    'Item' => [
        'title' => [
            'S' => $movieName,
        ],
        'year' => [
            'N' => $movieYear,
        ],
    ],
];
$attributes = ["rating" =>
    [
        'AttributeName' => 'rating',
        'AttributeType' => 'N',
        'Value' => $rating,
    ],
    'plot' => [
        'AttributeName' => 'plot',
        'AttributeType' => 'S',
        'Value' => $plot,
    ]
];
getService->updateItemAttributesByKey($tableName, $key, $attributes);
echo "Movie added and updated.";

$batch = json_decode(loadMovieData());

$limit = 0;
$service->writeBatch($tableName, $batch);
```

```
$movie = $service->getItemByKey($tableName, $key);
echo "\nThe movie {$movie['Item']['title']['S']} was released in
{$movie['Item']['year']['N']}.\n";
echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
$rating = 0;
while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 || $rating > 10) {
    $rating = testable_readline("Rating (1-10): ");
}
$service->updateItemAttributeByKey($tableName, $key, 'rating', 'N', $rating);

$movie = $service->getItemByKey($tableName, $key);
echo "Ok, you have rated {$movie['Item']['title']['S']} as a
{$movie['Item']['rating']['N']}\n";

$service->deleteItemByKey($tableName, $key);
echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

echo "That's okay though. The book was better. Now, for something lighter,
in what year were you born?\n";
$birthYear = "not a number";
while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
    $birthYear = testable_readline("Birth year: ");
}
$birthKey = [
    'Key' => [
        'year' => [
            'N' => "$birthYear",
        ],
    ],
];
$result = $service->query($tableName, $birthKey);
$marshal = new Marshaler();
echo "Here are the movies in our collection released the year you were
born:\n";
$oops = "Oops! There were no movies released in that year (that we know
of).\n";
$display = "";
foreach ($result['Items'] as $movie) {
    $movie = $marshal->unmarshalItem($movie);
    $display .= $movie['title'] . "\n";
}
echo ($display) ?: $oops;

$yearsKey = [
    'Key' => [
        'year' => [
            'N' => [
                'minRange' => 1990,
                'maxRange' => 1999,
            ],
        ],
    ],
];
$filter = "year between 1990 and 1999";
echo "\nHere's a list of all the movies released in the 90s:\n";
$result = $service->scan($tableName, $yearsKey, $filter);
foreach ($result['Items'] as $movie) {
    $movie = $marshal->unmarshalItem($movie);
    echo $movie['title'] . "\n";
}
```

```
        echo "\nCleaning up this demo by deleting table $tableName...\n";
        $service->deleteTable($tableName);
    }
}
```

- For API details, see the following topics in *AWS SDK for PHP API Reference*.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that encapsulates a DynamoDB table.

```
from decimal import Decimal
from io import BytesIO
import json
import logging
import os
from pprint import pprint
import requests
from zipfile import ZipFile
import boto3
from boto3.dynamodb.conditions import Key
from botocore.exceptions import ClientError
from question import Question

logger = logging.getLogger(__name__)

class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None

    def exists(self, table_name):
        """
        Determines whether a table exists. As a side effect, stores the table in
        a member variable.

        :param table_name: The name of the table to check.
        :return: True when the table exists; otherwise, False.
        """


```

```
"""
try:
    table = self.dyn_resource.Table(table_name)
    table.load()
    exists = True
except ClientError as err:
    if err.response['Error']['Code'] == 'ResourceNotFoundException':
        exists = False
    else:
        logger.error(
            "Couldn't check for existence of %s. Here's why: %s: %s",
            table_name,
            err.response['Error']['Code'], err.response['Error']
        )
raise
else:
    self.table = table
return exists

def create_table(self, table_name):
    """
    Creates an Amazon DynamoDB table that can be used to store movie data.
    The table uses the release year of the movie as the partition key and the
    title as the sort key.

    :param table_name: The name of the table to create.
    :return: The newly created table.
    """
    try:
        self.table = self.dyn_resource.create_table(
            TableName=table_name,
            KeySchema=[
                {'AttributeName': 'year', 'KeyType': 'HASH'}, # Partition key
                {'AttributeName': 'title', 'KeyType': 'RANGE'} # Sort key
            ],
            AttributeDefinitions=[
                {'AttributeName': 'year', 'AttributeType': 'N'},
                {'AttributeName': 'title', 'AttributeType': 'S'}
            ],
            ProvisionedThroughput={'ReadCapacityUnits': 10,
                                   'WriteCapacityUnits': 10})
        self.table.wait_until_exists()
    except ClientError as err:
        logger.error(
            "Couldn't create table %s. Here's why: %s: %s", table_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return self.table

def list_tables(self):
    """
    Lists the Amazon DynamoDB tables for the current account.

    :return: The list of tables.
    """
    try:
        tables = []
        for table in self.dyn_resource.tables.all():
            print(table.name)
            tables.append(table)
    except ClientError as err:
        logger.error(
            "Couldn't list tables. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
```

```
        else:
            return tables

    def write_batch(self, movies):
        """
        Fills an Amazon DynamoDB table with the specified data, using the Boto3
        Table.batch_writer() function to put the items in the table.
        Inside the context manager, Table.batch_writer builds a list of
        requests. On exiting the context manager, Table.batch_writer starts sending
        batches of write requests to Amazon DynamoDB and automatically
        handles chunking, buffering, and retrying.

        :param movies: The data to put in the table. Each item must contain at
        least
                           the keys required by the schema that was specified when the
                           table was created.
        """
        try:
            with self.table.batch_writer() as writer:
                for movie in movies:
                    writer.put_item(Item=movie)
        except ClientError as err:
            logger.error(
                "Couldn't load data into table %s. Here's why: %s: %s",
                self.table.name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise

    def add_movie(self, title, year, plot, rating):
        """
        Adds a movie to the table.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :param plot: The plot summary of the movie.
        :param rating: The quality rating of the movie.
        """
        try:
            self.table.put_item(
                Item={
                    'year': year,
                    'title': title,
                    'info': {'plot': plot, 'rating': Decimal(str(rating))}})
        except ClientError as err:
            logger.error(
                "Couldn't add movie %s to table %s. Here's why: %s: %s",
                title, self.table.name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise

    def get_movie(self, title, year):
        """
        Gets movie data from the table for a specific movie.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :return: The data about the requested movie.
        """
        try:
            response = self.table.get_item(Key={'year': year, 'title': title})
        except ClientError as err:
            logger.error(
                "Couldn't get movie %s from table %s. Here's why: %s: %s",
                title, self.table.name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

```

        else:
            return response['Item']

    def update_movie(self, title, year, rating, plot):
        """
        Updates rating and plot data for a movie in the table.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating: The updated rating to give the movie.
        :param plot: The updated plot summary to give the movie.
        :return: The fields that were updated, with their new values.
        """
        try:
            response = self.table.update_item(
                Key={'year': year, 'title': title},
                UpdateExpression="set info.rating=:r, info.plot=:p",
                ExpressionAttributeValues={
                    ':r': Decimal(str(rating)), ':p': plot},
                ReturnValues="UPDATED_NEW")
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title, self.table.name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['Attributes']

    def query_movies(self, year):
        """
        Queries for movies that were released in the specified year.

        :param year: The year to query.
        :return: The list of movies that were released in the specified year.
        """
        try:
            response =
self.table.query(KeyConditionExpression=Key('year').eq(year))
        except ClientError as err:
            logger.error(
                "Couldn't query for movies released in %s. Here's why: %s: %s",
                year,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['Items']

    def scan_movies(self, year_range):
        """
        Scans for movies that were released in a range of years.
        Uses a projection expression to return a subset of data for each movie.

        :param year_range: The range of years to retrieve.
        :return: The list of movies released in the specified years.
        """
        movies = []
        scan_kwargs = {
            'FilterExpression': Key('year').between(year_range['first'],
year_range['second']),
            'ProjectionExpression': "#yr, title, info.rating",
            'ExpressionAttributeNames': {"#yr": "year"}}
        try:
            done = False
            start_key = None
            while not done:

```

```
        if start_key:
            scan_kwargs['ExclusiveStartKey'] = start_key
            response = self.table.scan(**scan_kwargs)
            movies.extend(response.get('Items', []))
            start_key = response.get('LastEvaluatedKey', None)
            done = start_key is None
        except ClientError as err:
            logger.error(
                "Couldn't scan for movies. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise

    return movies

def delete_movie(self, title, year):
    """
    Deletes a movie from the table.

    :param title: The title of the movie to delete.
    :param year: The release year of the movie to delete.
    """
    try:
        self.table.delete_item(Key={'year': year, 'title': title})
    except ClientError as err:
        logger.error(
            "Couldn't delete movie %s. Here's why: %s: %s",
            title, err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def delete_table(self):
    """
    Deletes the table.
    """
    try:
        self.table.delete()
        self.table = None
    except ClientError as err:
        logger.error(
            "Couldn't delete table. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
```

Create a helper function to download and extract the sample JSON file.

```
def get_sample_movie_data(movie_file_name):
    """
    Gets sample movie data, either from a local file or by first downloading it
    from
    the Amazon DynamoDB developer guide.

    :param movie_file_name: The local file name where the movie data is stored in
    JSON format.
    :return: The movie data as a dict.
    """
    if not os.path.isfile(movie_file_name):
        print(f"Downloading {movie_file_name}...")
        movie_content = requests.get(
            'https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
samples/moviedata.zip')
        movie_zip = ZipFile(BytesIO(movie_content.content))
        movie_zip.extractall()

    try:
        with open(movie_file_name) as movie_file:
```

```
        movie_data = json.load(movie_file, parse_float=Decimal)
    except FileNotFoundError:
        print(f"File {movie_file_name} not found. You must first download the file
to "
              "run this demo. See the README for instructions.")
        raise
    else:
        # The sample file lists over 4000 movies, return only the first 250.
        return movie_data[:250]
```

Run an interactive scenario to create the table and perform actions on it.

```
def run_scenario(table_name, movie_file_name, dyn_resource):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the Amazon DynamoDB getting started demo.")
    print('*'*88)

    movies = Movies(dyn_resource)
    movies_exists = movies.exists(table_name)
    if not movies_exists:
        print(f"\nCreating table {table_name}...")
        movies.create_table(table_name)
        print(f"\nCreated table {movies.table.name}.")"

    my_movie = Question.ask_questions([
        Question('title', "Enter the title of a movie you want to add to the table:"),
        Question('year', "What year was it released?", Question.is_int),
        Question(
            'rating', "On a scale of 1 - 10, how do you rate it?",
            Question.is_float, Question.in_range(1, 10)),
        Question('plot', "Summarize the plot for me:")
    ])
    movies.add_movie(**my_movie)
    print(f"\nAdded '{my_movie['title']}' to '{movies.table.name}'")
    print('*'*88)

    movie_update = Question.ask_questions([
        Question(
            'rating',
            f"\nLet's update your movie.\nYou rated it {my_movie['rating']}, what
new "
            f"rating would you give it?", Question.is_float, Question.in_range(1,
10)),
        Question(
            'plot',
            f"You summarized the plot as '{my_movie['plot']}'.\nWhat would you say
now?")]
    )
    my_movie.update(movie_update)
    updated = movies.update_movie(**my_movie)
    print(f"\nUpdated '{my_movie['title']}' with new attributes:")
    pprint(updated)
    print('*'*88)

    if not movies_exists:
        movie_data = get_sample_movie_data(movie_file_name)
        print(f"\nReading data from '{movie_file_name}' into your table.")
        movies.write_batch(movie_data)
        print(f"\nWrote {len(movie_data)} movies into {movies.table.name}.")
    print('*'*88)

    title = "The Lord of the Rings: The Fellowship of the Ring"
```

```

if Question.ask_question(
    f"Let's move on...do you want to get info about '{title}'? (y/n) ",
    Question.is_yesno):
    movie = movies.get_movie(title, 2001)
    print("\nHere's what I found:")
    pprint(movie)
print('*88)

ask_for_year = True
while ask_for_year:
    release_year = Question.ask_question(
        f"\nLet's get a list of movies released in a given year. Enter a year
between "
        f"1972 and 2018: ", Question.is_int, Question.in_range(1972, 2018))
    releases = movies.query_movies(release_year)
    if releases:
        print(f"There were {len(releases)} movies released in {release_year}:")
        for release in releases:
            print(f"\t{release['title']}")
        ask_for_year = False
    else:
        print(f"I don't know about any movies released in {release_year}!")
    ask_for_year = Question.ask_question("Try another year? (y/n) ",
    Question.is_yesno)
print('*88)

years = Question.ask_questions([
    Question(
        'first',
        f"\nNow let's scan for movies released in a range of years. Enter a
year: ",
        Question.is_int, Question.in_range(1972, 2018)),
    Question(
        'second', "Now enter another year: ",
        Question.is_int, Question.in_range(1972, 2018)))
releases = movies.scan_movies(years)
if releases:
    count = Question.ask_question(
        f"\nFound {len(releases)} movies. How many do you want to see? ",
        Question.is_int, Question.in_range(1, len(releases)))
    print(f"\nHere are your {count} movies:\n")
    pprint(releases[:count])
else:
    print(f"I don't know about any movies released between {years['first']} "
          f"and {years['second']}.")
print('*88)

if Question.ask_question(
    f"\nLet's remove your movie from the table. Do you want to remove "
    f"'{my_movie['title']}'? (y/n)", Question.is_yesno):
    movies.delete_movie(my_movie['title'], my_movie['year'])
    print(f"\nRemoved '{my_movie['title']}' from the table.")
print('*88)

if Question.ask_question(f"\nDelete the table? (y/n) ", Question.is_yesno):
    movies.delete_table()
    print(f"Deleted {table_name}.")
else:
    print("Don't forget to delete the table when you're done or you might incur
"
          "charges on your account.")

print("\nThanks for watching!")
print('*88)

```

```
if __name__ == '__main__':
    try:
        run_scenario(
            'doc-example-table-movies', 'moviedata.json',
            boto3.resource('dynamodb'))
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

This scenario uses the following helper class to ask questions at a command prompt.

```
class Question:
    """
    A helper class to ask questions at a command prompt and validate and convert
    the answers.
    """
    def __init__(self, key, question, *validators):
        """
        :param key: The key that is used for storing the answer in a dict, when
                   multiple questions are asked in a set.
        :param question: The question to ask.
        :param validators: The answer is passed through the list of validators
        until
                           one fails or they all pass. Validators may also convert
        the
                           answer to another form, such as from a str to an int.
        """
        self.key = key
        self.question = question
        self.validators = Question.non_empty, *validators

    @staticmethod
    def ask_questions(questions):
        """
        Asks a set of questions and stores the answers in a dict.

        :param questions: The list of questions to ask.
        :return: A dict of answers.
        """
        answers = {}
        for question in questions:
            answers[question.key] = Question.ask_question(
                question.question, *question.validators)
        return answers

    @staticmethod
    def ask_question(question, *validators):
        """
        Asks a single question and validates it against a list of validators.
        When an answer fails validation, the complaint is printed and the question
        is asked again.

        :param question: The question to ask.
        :param validators: The list of validators that the answer must pass.
        :return: The answer, converted to its final form by the validators.
        """
        answer = None
        while answer is None:
            answer = input(question)
            for validator in validators:
                answer, complaint = validator(answer)
                if answer is None:
                    print(complaint)
                    break
        return answer
```

```
@staticmethod
def non_empty(answer):
    """
    Validates that the answer is not empty.
    :return: The non-empty answer, or None.
    """
    return answer if answer != '' else None, "I need an answer. Please?"

@staticmethod
def is_yesno(answer):
    """
    Validates a yes/no answer.
    :return: True when the answer is 'y'; otherwise, False.
    """
    return answer.lower() == 'y', ""

@staticmethod
def is_int(answer):
    """
    Validates that the answer can be converted to an int.
    :return: The int answer; otherwise, None.
    """
    try:
        int_answer = int(answer)
    except ValueError:
        int_answer = None
    return int_answer, f"{answer} must be a valid integer."

@staticmethod
def is_letter(answer):
    """
    Validates that the answer is a letter.
    :return: The letter answer, converted to uppercase; otherwise, None.
    """
    return answer.upper() if answer.isalpha() else None, f"{answer} must be a single letter."

@staticmethod
def is_float(answer):
    """
    Validate that the answer can be converted to a float.
    :return: The float answer; otherwise, None.
    """
    try:
        float_answer = float(answer)
    except ValueError:
        float_answer = None
    return float_answer, f"{answer} must be a valid float."

@staticmethod
def in_range(lower, upper):
    """
    Validate that the answer is within a range. The answer must be of a type
    that can
    be compared to the lower and upper bounds.
    :return: The answer, if it is within the range; otherwise, None.
    """
    def _validate(answer):
        return (
            answer if lower <= answer <= upper else None,
            f"{answer} must be between {lower} and {upper}.")
    return _validate
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that encapsulates a DynamoDB table.

```
require "aws-sdk-dynamodb"
require "json"
require "open-uri"
require "pp"
require "zip"
require_relative "question"

# Encapsulates an Amazon DynamoDB table of movie data.
class Movies
  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(dynamo_resource)
    @dynamo_resource = dynamo_resource
  end

  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
  def exists?(table_name)
    table = Aws::DynamoDB::Table.new(table_name)
    table.load
    @table = table
    rescue Aws::DynamoDB::Errors::ResourceNotFoundException
      puts("Table #{table_name} doesn't exist. Let's create it.")
      false
    rescue Aws::Errors::ServiceError => e
      puts("Couldn't check for existence of #{table_name}. Here's why:")
      puts("\t#{e.code}: #{e.message}")
      raise
    else
      !@table.nil?
    end

    # Creates an Amazon DynamoDB table that can be used to store movie data.
    # The table uses the release year of the movie as the partition key and the
```

```
# title as the sort key.  
#  
# @param table_name [String] The name of the table to create.  
# @return [Aws::DynamoDB::Table] The newly created table.  
def create_table(table_name)  
  @table = @dynamo_resource.create_table(  
    table_name: table_name,  
    key_schema: [  
      {attribute_name: "year", key_type: "HASH"}, # Partition key  
      {attribute_name: "title", key_type: "RANGE"} # Sort key  
    ],  
    attribute_definitions: [  
      {attribute_name: "year", attribute_type: "N"},  
      {attribute_name: "title", attribute_type: "S"}  
    ],  
    provisioned_throughput: {read_capacity_units: 10, write_capacity_units: 10})  
  @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)  
rescue Aws::Errors::ServiceError => e  
  puts("Couldn't create table #{table_name}. Here's why:")  
  puts("\t#{e.code}: #{e.message}")  
  raise  
else  
  @table  
end  
  
# Fills an Amazon DynamoDB table with the specified data. Items are sent in  
# batches of 25 until all items are written.  
#  
# @param movies [Enumerable] The data to put in the table. Each item must contain  
at least  
#                                     the keys required by the schema that was specified  
when the  
#                                     table was created.  
def write_batch(movies)  
  index = 0  
  slice_size = 25  
  while index < movies.length  
    movie_items = []  
    movies[index, slice_size].each do |movie|  
      movie_items.append({put_request: {item: movie}})  
    end  
    @dynamo_resource.batch_write_item({request_items: {@table.name =>  
      movie_items}})  
    index += slice_size  
  end  
rescue Aws::Errors::ServiceError => e  
  puts(  
    "Couldn't load data into table #{@table.name}. Here's why:")  
  puts("\t#{e.code}: #{e.message}")  
  raise  
end  
  
# Adds a movie to the table.  
#  
# @param title [String] The title of the movie.  
# @param year [Integer] The release year of the movie.  
# @param plot [String] The plot summary of the movie.  
# @param rating [Float] The quality rating of the movie.  
def add_movie(title:, year:, plot:, rating:)  
  @table.put_item(  
    item: {  
      "year" => year,  
      "title" => title,  
      "info" => {"plot" => plot, "rating" => rating}})  
rescue Aws::Errors::ServiceError => e  
  puts("Couldn't add movie #{title} to table #{@table.name}. Here's why:")
```

```

    puts("\t#{e.code}: #{e.message}")
    raise
end

# Gets movie data from the table for a specific movie.
#
# @param title [String] The title of the movie.
# @param year [Integer] The release year of the movie.
# @return [Hash] The data about the requested movie.
def get_movie(title, year)
  response = @table.get_item(key: {"year" => year, "title" => title})
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get movie #{title} from table #{@table.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  response.item
end

# Updates rating and plot data for a movie in the table.
#
# @param title [String] The title of the movie to update.
# @param year [Int] The release year of the movie to update.
# @param rating [Float] The updated rating to give the movie.
# @param plot [String] The updated plot summary to give the movie.
# @return [Hash] The fields that were updated, with their new values.
def update_movie(title:, year:, rating:, plot:)
  response = @table.update_item(
    key: {"year" => year, "title" => title},
    update_expression: "set info.rating=:r, info.plot=:p",
    expression_attribute_values: { ":r" => rating, ":p" => plot },
    return_values: "UPDATED_NEW")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't update movie #{title} in table #{@table.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  response.attributes
end

# Queries for movies that were released in the specified year.
#
# @param year [Integer] The year to query.
# @return [Array] The list of movies that were released in the specified year.
def query_movies(year)
  response = @table.query(
    key_condition_expression: "#yr = :year",
    expression_attribute_names: {"#yr" => "year"},
    expression_attribute_values: {":year" => year})
rescue Aws::Errors::ServiceError => e
  puts("Couldn't query for movies released in #{year}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  response.items
end

# Scans for movies that were released in a range of years.
# Uses a projection expression to return a subset of data for each movie.
#
# @param year_range [Hash] The range of years to retrieve.
# @return [Array] The list of movies released in the specified years.
def scan_movies(year_range)
  movies = []
  scan_hash = {
    filter_expression: "#yr between :start_yr and :end_yr",

```

```

projection_expression: "#yr, title, info.rating",
expression_attribute_names: {"#yr" => "year"},
expression_attribute_values: {
    ":start_yr" => year_range[:start], ":end_yr" => year_range[:end]}
}
done = false
start_key = nil
until done
    scan_hash[:exclusive_start_key] = start_key unless start_key.nil?
    response = @table.scan(scan_hash)
    movies.concat(response.items) unless response.items.nil?
    start_key = response.last_evaluated_key
    done = start_key.nil?
end
rescue Aws::Errors::ServiceError => e
    puts("Couldn't scan for movies. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    movies
end

# Deletes a movie from the table.
#
# @param title [String] The title of the movie to delete.
# @param year [Integer] The release year of the movie to delete.
def delete_movie(title, year)
    @table.delete_item(key: {"year" => year, "title" => title})
rescue Aws::Errors::ServiceError => e
    puts("Couldn't delete movie #{title}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
end

# Deletes the table.
def delete_table
    @table.delete
    @table = nil
rescue Aws::Errors::ServiceError => e
    puts("Couldn't delete table. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
end
end

```

Create a helper function to download and extract the sample JSON file.

```

# Gets sample movie data, either from a local file or by first downloading it from
# the Amazon DynamoDB Developer Guide.
#
# @param movie_file_name [String] The local file name where the movie data is
# stored in JSON format.
# @return [Hash] The movie data as a Hash.
def get_sample_movie_data(movie_file_name)
    if !File.file?(movie_file_name)
        puts("Downloading #{movie_file_name}...")
        movie_content = URI.open(
            "https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/
moviedata.zip"
        )
        movie_json = ""
        Zip::File.open_buffer(movie_content) do |zip|
            zip.each do |entry|
                movie_json = entry.get_input_stream.read

```

```

        end
      end
    else
      movie_json = File.read(movie_file_name)
    end
    movie_data = JSON.parse(movie_json)
    # The sample file lists over 4000 movies. This returns only the first 250.
    movie_data.slice(0, 250)
rescue Errno::ENOENT
  puts("File #{movie_file_name} not found. Before you can run this demo, you must
  \"\
    download the file. For instructions, see the README.\")
  raise
end

```

Run an interactive scenario to create the table and perform actions on it.

```

# Runs the DynamoDB getting started demo.
#
# @param movies [Movies] A wrapper class initialized with a DynamoDB resource.
# @param table_name [String] The name to give the movie table.
# @param movie_file_name [String] The name of a file that contains movie data in
#   JSON
#                               format. This data is loaded into the movie table
#   as part of the demo.
def run_scenario(movies, table_name, movie_file_name)
  puts("-" * 88)
  puts("Welcome to the DynamoDB getting started demo.")
  puts("-" * 88)

  movies_exists = movies.exists?(table_name)
  unless movies_exists
    puts("\nCreating table #{table_name}...")
    movies.create_table(table_name)
    puts("\nCreated table #{movies.table.name}.")
  end

  my_movie = {}
  my_movie[:title] = Question.ask("Enter the title of a movie to add to the table:
")
  my_movie[:year] = Question.ask("What year was it released? ", method(:is_int))
  my_movie[:rating] = Question.ask(
    "On a scale of 1 - 10, how do you rate it? ", method(:is_float), in_range(1,
10)
  )
  my_movie[:plot] = Question.ask("Summarize the plot for me: ")

  movies.add_movie(**my_movie)
  puts("\nAdded '#{my_movie[:title]}' to '#{movies.table.name}'")
  puts("-" * 88)

  puts("Let's update your movie. You rated it #{my_movie[:rating]}")
  my_movie[:rating] = Question.ask("What new rating would you give it? ",
method(:is_float), in_range(1, 10))
  puts("You summarized the plot as '#{my_movie[:plot]}'.")
  my_movie[:plot] = Question.ask("What would you say now? ")
  updated = movies.update_movie(**my_movie)
  puts("Updated '#{my_movie[:title]}' with new attributes:")
  pp(updated)
  puts("-" * 88)

  unless movies_exists
    movie_data = get_sample_movie_data(movie_file_name)
    puts("Reading data from '#{movie_file_name}' into your table.")
  end

```

```
movies.write_batch(movie_data)
puts("Wrote #{movie_data.length} movies into #{movies.table.name}.")
puts("-" * 88)
end

title = "The Lord of the Rings: The Fellowship of the Ring"
if Question.ask("Let's move on. Do you want to get info about '#{title}'? (y/n)",
",
    method(:is_yesno))
movie = movies.get_movie(title, 2001)
puts("\nHere's what I found:")
pp(movie)
puts("-" * 88)
end

ask_for_year = true
puts("Let's get a list of movies released in a given year.")
while ask_for_year
    release_year = Question.ask(
        "Enter a year between 1972 and 2018: ", method(:is_int), in_range(1972,
2018))
    releases = movies.query_movies(release_year)
    if !releases.empty?
        puts("There were #{releases.length} movies released in #{release_year}:")
        releases.each do |release|
            puts("\t#{release['title']}")  

            ask_for_year = false
        end
    else
        puts("I don't know about any movies released in #{release_year}!")
        ask_for_year = Question.ask("Try another year? (y/n) ", method(:is_yesno))
        puts("-" * 88)
    end
end

years = {}
years[:start] = Question.ask(
    "Let's scan for movies released in a range of years. Enter a year: ",
    method(:is_int), in_range(1972, 2018))
years[:end] = Question.ask(
    "Now enter another year: ", method(:is_int), in_range(1972, 2018))
releases = movies.scan_movies(years)
if !releases.empty?
    puts("Found #{releases.length} movies.")
    count = Question.ask(
        "How many do you want to see? ", method(:is_int), in_range(1,
releases.length))
    puts("Here are your #{count} movies:")
    releases.take(count).each do |release|
        puts("\t#{release['title']}")  

    end
else
    puts("I don't know about any movies released between #{years[:start]} \"\
        and #{years[:end]}\"")
    puts("-" * 88)
end

puts("Let's remove your movie from the table.")
if Question.ask(
    "Do you want to remove '#{my_movie[:title]}'? (y/n) ", method(:is_yesno))
    movies.delete_movie(my_movie[:title], my_movie[:year])
    puts("Removed '#{my_movie[:title]}' from the table.")
    puts("-" * 88)
end

if Question.ask("Delete the table? (y/n) ", method(:is_yesno))
```

```
    movies.delete_table
    puts("Deleted #{table_name}.")
else
  puts("Don't forget to delete the table when you're done or you might incur \"\
    charges on your account.")
end

puts("\nThanks for watching!")
puts("-" * 88)
rescue Aws::Errors::ServiceError
  puts("Something went wrong with the demo.")
rescue Errno::ENOENT
  true
end

run_scenario(
  Movies.new(Aws::DynamoDB::Resource.new),
  "doc-example-table-movies", "moviedata.json",
) if $PROGRAM_NAME == __FILE__
```

This scenario uses the following helper class to ask questions at a command prompt.

```
# Asks a single question and validates it against a list of validators.
# When an answer fails validation, the complaint is printed and the question
# is asked again.
#
# @param question [String] The question to ask.
# @param validators [Array] The list of validators that the answer must pass.
# @return The answer, converted to its final form by the validators.
class Question
  def self.ask(question, *validators)
    answer = nil
    while answer.nil?
      puts(question)
      answer = gets.chomp
      validators.unshift(method(:non_empty)) unless validators[0] ==
method(:non_empty)
      validators.each do |validator|
        answer, complaint = validator.call(answer)
        if answer.nil?
          puts(complaint)
          break
        end
      end
    end
    answer
  end
end

# Validates that the answer is not empty.
# @return [Array] The non-empty answer, or nil.
def non_empty(answer)
  answer = nil unless answer != ""
  [answer, "I need an answer. Please?"]
end

# Validates a yes/no answer.
# @return [Array] True when the answer is 'y'; otherwise, False.
def is_yesno(answer)
  [answer.downcase == "y", ""]
end

# Validates that the answer can be converted to an int.
# @return [Array] The int answer; otherwise, nil.
```

```
def is_int(answer)
  int_answer = answer.to_i
  if int_answer == 0
    int_answer = nil
  end
  [int_answer, "#{answer} must be a valid integer."]
end

# Validates that the answer can be converted to a float.
# :return [Array] The float answer; otherwise, None.
def is_float(answer)
  float_answer = answer.to_f
  if float_answer == 0.0
    float_answer = nil
  end
  [float_answer, "#{answer} must be a valid float."]
end

# Validates that the answer is within a range. The answer must be of a type that
# can
# be compared to the lower and upper bounds.
# @return [Proc] A Proc that can be called to determine whether the answer is
# within
#           the expected range.
def in_range(lower, upper)
  Proc.new { |answer|
    answer.between?(lower, upper) ? range_answer = answer : range_answer = nil
    [range_answer, "#{answer} must be between #{lower} and #{upper}."]
  }
end
```

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## Query a DynamoDB table by using batches of PartiQL statements and an AWS SDK

The following code examples show how to:

- Get a batch of items by running multiple SELECT statements.
- Add a batch of items by running multiple INSERT statements.
- Update a batch of items by running multiple UPDATE statements.
- Delete a batch of items by running multiple DELETE statements.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Before you run this example, download 'movies.json' from
// https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
GettingStarted.Js.02.html,
// and put it in the same folder as the example.

// Separator for the console display.
var SepBar = new string('-', 80);
const string tableName = "movie_table";
const string movieFileName = "moviedata.json";

DisplayInstructions();

// Create the table and wait for it to be active.
Console.WriteLine($"Creating the movie table: {tableName}");

var success = await DynamoDBMethods.CreateMovieTableAsync(tableName);
if (success)
{
    Console.WriteLine($"Successfully created table: {tableName}.");
}

WaitForEnter();

// Add movie information to the table from moviedata.json. See the
// instructions at the top of this file to download the JSON file.
Console.WriteLine($"Inserting movies into the new table. Please wait...");
success = await PartiQLBatchMethods.InsertMovies(tableName, movieFileName);
if (success)
{
    Console.WriteLine("Movies successfully added to the table.");
}
else
{
    Console.WriteLine("Movies could not be added to the table.");
}

WaitForEnter();

// Update multiple movies by using the BatchExecute statement.
var title1 = "Star Wars";
var year1 = 1977;
var title2 = "Wizard of Oz";
var year2 = 1939;

Console.WriteLine($"Updating two movies with producer information: {title1} and
{title2}.");
success = await PartiQLBatchMethods.GetBatch(tableName, title1, title2, year1,
year2);
if (success)
{
    Console.WriteLine($"Successfully retrieved {title1} and {title2}.");
}
else
{
    Console.WriteLine("Select statement failed.");
```

```
}

WaitForEnter();

// Update multiple movies by using the BatchExecute statement.
var producer1 = "LucasFilm";
var producer2 = "MGM";

Console.WriteLine($"Updating two movies with producer information: {title1} and {title2}.");
success = await PartiQLBatchMethods.UpdateBatch(tableName, producer1, title1, year1, producer2, title2, year2);
if (success)
{
    Console.WriteLine($"Successfully updated {title1} and {title2}.");
}
else
{
    Console.WriteLine("Update failed.");
}

WaitForEnter();

// Delete multiple movies by using the BatchExecute statement.
Console.WriteLine($"Now we will delete {title1} and {title2} from the table.");
success = await PartiQLBatchMethods.DeleteBatch(tableName, title1, year1, title2, year2);

if (success)
{
    Console.WriteLine($"Deleted {title1} and {title2}");
}
else
{
    Console.WriteLine($"could not delete {title1} or {title2}");
}

WaitForEnter();

// DNow that the PartiQL Batch scenario is complete, delete the movie table.
success = await DynamoDBMethods.DeleteTableAsync(tableName);

if (success)
{
    Console.WriteLine($"Successfully deleted {tableName}");
}
else
{
    Console.WriteLine($"Could not delete {tableName}");
}

/// <summary>
/// Displays the description of the application on the console.
/// </summary>
void DisplayInstructions()
{
    Console.Clear();
    Console.WriteLine();
    Console.Write(new string(' ', 24));
    Console.WriteLine("Dynamodb PartiQL Basics Example");
    Console.WriteLine(SepBar);
    Console.WriteLine("This demo application shows the basics of using Amazon");
    DynamoDB with the AWS SDK for");
    Console.WriteLine(".NET version 3.7 and .NET 6.");
    Console.WriteLine(SepBar);
    Console.WriteLine("Creates a table by using the CreateTable method.");
}
```

```
Console.WriteLine("Gets multiple movies by using a PartiQL SELECT statement.");
Console.WriteLine("Updates multiple movies by using the ExecuteBatch method.");
Console.WriteLine("Deletes multiple movies by using a PartiQL DELETE
statement.");
Console.WriteLine("Cleans up the resources created for the demo by deleting the
table.");
Console.WriteLine(SepBar);

WaitForEnter();
}


Simple method to wait for the <Enter> key to be pressed.

void WaitForEnter()
{
    Console.WriteLine("\nPress <Enter> to continue.");
    Console.Write(SepBar);
    _ = Console.ReadLine();
}



public static async Task<bool> GetBatch(
    string tableName,
    string title1,
    string title2,
    int year1,
    int year2)
{
    var getBatch = $"SELECT FROM {tableName} WHERE title = ? AND year = ?";
    var statements = new List<BatchStatementRequest>
    {
        new BatchStatementRequest
        {
            Statement = getBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = title1 },
                new AttributeValue { N = year1.ToString() },
            },
        },
        new BatchStatementRequest
        {
            Statement = getBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = title2 },
                new AttributeValue { N = year2.ToString() },
            },
        }
    };

    var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
{
```

```
        Statements = statements,
    });

    if (response.Responses.Count > 0)
    {
        response.Responses.ForEach(r =>
        {
            Console.WriteLine($"{r.Item["title"]}\t{r.Item["year"]}");
        });
        return true;
    }
    else
    {
        Console.WriteLine($"Couldn't find either {title1} or {title2}.");
        return false;
    }
}

/// <summary>
/// Inserts movies imported from a JSON file into the movie table by
/// using an Amazon DynamoDB PartiQL INSERT statement.
/// </summary>
/// <param name="tableName">The name of the table into which the movie
/// information will be inserted.</param>
/// <param name="movieFileName">The name of the JSON file that contains
/// movie information.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the insert operation.</returns>
public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
{
    // Get the list of movies from the JSON file.
    var movies = ImportMovies(movieFileName);

    var success = false;

    if (movies is not null)
    {
        // Insert the movies in a batch using PartiQL. Because the
        // batch can contain a maximum of 25 items, insert 25 movies
        // at a time.
        string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?, 'year': ?}}";
        var statements = new List<BatchStatementRequest>();

        try
        {
            for (var indexOffset = 0; indexOffset < 250; indexOffset += 25)
            {
                for (var i = indexOffset; i < indexOffset + 25; i++)
                {
                    statements.Add(new BatchStatementRequest
                    {
                        Statement = insertBatch,
                        Parameters = new List<AttributeValue>
                        {
                            new AttributeValue { S = movies[i].Title },
                            new AttributeValue { N =
movies[i].Year.ToString() },
                        },
                    });
                }
            }
        
```

```

        var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
{
    Statements = statements,
});

// Wait between batches for movies to be successfully
added.
System.Threading.Thread.Sleep(3000);

success = response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;

// Clear the list of statements for the next batch.
statements.Clear();
}

}
catch (AmazonDynamoDBException ex)
{
    Console.WriteLine(ex.Message);
}
}

return success;
}

/// <summary>
/// Loads the contents of a JSON file into a list of movies to be
/// added to the DynamoDB table.
/// </summary>
/// <param name="movieFileName">The full path to the JSON file.</param>
/// <returns>A generic list of movie objects.</returns>
public static List<Movie> ImportMovies(string movieFileName)
{
    if (!File.Exists(movieFileName))
    {
        return null;
    }

    using var sr = new StreamReader(movieFileName);
    string json = sr.ReadToEnd();
    var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

    if (allMovies is not null)
    {
        // Return the first 250 entries.
        return allMovies.GetRange(0, 250);
    }
    else
    {
        return null;
    }
}

/// <summary>
/// Updates information for multiple movies.
/// </summary>
/// <param name="tableName">The name of the table containing the
/// movies to be updated.</param>
/// <param name="producer1">The producer name for the first movie
/// to update.</param>
/// <param name="title1">The title of the first movie.</param>
/// <param name="year1">The year that the first movie was released.</param>
/// <param name="producer2">The producer name for the second

```

```

    ///> movie to update.</param>
    ///> <param name="title2">The title of the second movie.</param>
    ///> <param name="year2">The year that the second movie was released.</param>
    ///> <returns>A Boolean value that indicates the success of the update.</returns>
public static async Task<bool> UpdateBatch(
    string tableName,
    string producer1,
    string title1,
    int year1,
    string producer2,
    string title2,
    int year2)
{
    string updateBatch = $"UPDATE {tableName} SET Producer=? WHERE title = ? AND year = ?";
    var statements = new List<BatchStatementRequest>
    {
        new BatchStatementRequest
        {
            Statement = updateBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = producer1 },
                new AttributeValue { S = title1 },
                new AttributeValue { N = year1.ToString() },
            },
        },
        new BatchStatementRequest
        {
            Statement = updateBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = producer2 },
                new AttributeValue { S = title2 },
                new AttributeValue { N = year2.ToString() },
            },
        }
    };

    var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
{
    Statements = statements,
});

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

    ///> summary>
    ///> Deletes multiple movies using a PartiQL BatchExecuteAsync
    ///> statement.
    ///> </summary>
    ///> <param name="tableName">The name of the table containing the
    ///> moves that will be deleted.</param>
    ///> <param name="title1">The title of the first movie.</param>
    ///> <param name="year1">The year the first movie was released.</param>
    ///> <param name="title2">The title of the second movie.</param>
    ///> <param name="year2">The year the second movie was released.</param>
    ///> <returns>A Boolean value indicating the success of the operation.</returns>

```

```
public static async Task<bool> DeleteBatch(
    string tableName,
    string title1,
    int year1,
    string title2,
    int year2)
{
    string updateBatch = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";
    var statements = new List<BatchStatementRequest>
    {
        new BatchStatementRequest
        {
            Statement = updateBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = title1 },
                new AttributeValue { N = year1.ToString() },
            },
        },
        new BatchStatementRequest
        {
            Statement = updateBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = title2 },
                new AttributeValue { N = year2.ToString() },
            },
        }
    };

    var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
{
    Statements = statements,
});

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
// 1. Create a table. (CreateTable)
if
(AwsDoc::DynamoDB::createDynamoDBTable(AwsDoc::DynamoDB::MOVIE_TABLE_NAME,
                                         clientConfig)) {

    AwsDoc::DynamoDB::partiqlBatchExecuteScenario(clientConfig);
```

```

        // 7. Delete the table. (DeleteTable)
        AwsDoc::DynamoDB::deleteDynamoTable(AwsDoc::DynamoDB::MOVIE_TABLE_NAME,
                                              clientConfig);
    }

//! Scenario to modify and query a DynamoDB table using PartiQL batch statements.
/*!!
 * \sa partiqlBatchExecuteScenario()
 * \param clientConfiguration: Aws client configuration.
 * \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::partiqlBatchExecuteScenario(
    const Aws::Client::ClientConfiguration &clientConfiguration) {

    // 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::vector<Aws::String> titles;
    std::vector<float> ratings;
    std::vector<int> years;
    std::vector<Aws::String> plots;
    Aws::String doAgain = "n";
    do {
        Aws::String aTitle = askQuestion(
            "Enter the title of a movie you want to add to the table: ");
        titles.push_back(aTitle);
        int aYear = askQuestionForInt("What year was it released? ");
        years.push_back(aYear);
        float aRating = askQuestionForFloatRange(
            "On a scale of 1 - 10, how do you rate it? ",
            1, 10);
        ratings.push_back(aRating);
        Aws::String aPlot = askQuestion("Summarize the plot for me: ");
        plots.push_back(aPlot);

        doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/n)"));
    } while (doAgain == "y");

    std::cout << "Adding " << titles.size()
        << (titles.size() == 1 ? " movie " : " movies ")
        << "to the table using a batch \"INSERT\" statement." << std::endl;
}

{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());

    std::stringstream sqlStream;
    sqlStream << "INSERT INTO \""
        << MOVIE_TABLE_NAME << "\" VALUE {\""
        << TITLE_KEY << ": ?, '" << YEAR_KEY << ": ?, '"
        << INFO_KEY << ": ?}";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

        // Create attribute for the info map.
        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

```

```

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
    Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
        ALLOCATION_TAG.c_str());
    ratingAttribute->SetN(ratings[i]);
    infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
    Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
        ALLOCATION_TAG.c_str());
    plotAttribute->SetS(plots[i]);
    infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
    attributes.push_back(infoMapAttribute);
    statements[i].SetParameters(attributes);
}

Aws::DynamoDB::Model::BatchExecuteStatementRequest request;
request.SetStatements(statements);

Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
    request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to add the movies: " <<
outcome.GetError().GetMessage()
    << std::endl;
    return false;
}
std::cout << "Retrieving the movie data with a batch \"SELECT\" statement."
    << std::endl;

// 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM `"
    << MOVIE_TABLE_NAME << "` WHERE "
    << TITLE_KEY << "=?" and " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;
    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
    request);
    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

```

```

        const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

        for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
            const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

            printMovieInfo(item);
        }
    }
    else {
        std::cerr << "Failed to retrieve the movie information: "
             << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

// 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

for (size_t i = 0; i < titles.size(); ++i) {
    ratings[i] = askQuestionForFloatRange(
        Aws::String("\nLet's update your the movie, \\"") + titles[i] +
        ".\nYou rated it " + std::to_string(ratings[i]) +
        ", what new rating would you give it? ", 1, 10);
}

std::cout << "Updating the movie with a batch \"UPDATE\" statement." <<
std::endl;

{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());

    std::stringstream sqlStream;
    sqlStream << "UPDATE \" " << MOVIE_TABLE_NAME << "\" SET "
             << INFO_KEY << "." << RATING_KEY << "=? WHERE "
             << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);
    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to update movie information: "
             << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

```

```

        }

        std::cout << "Retrieving the updated movie data with a batch \"SELECT\" statement."
                     << std::endl;

        // 5. Get the updated data for multiple movies using "Select" statements.
        (BatchExecuteStatement)
        {
            Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
            std::stringstream sqlStream;
            sqlStream << "SELECT * FROM \\" << MOVIE_TABLE_NAME << "\\" WHERE "
                         << TITLE_KEY << "=?" and " << YEAR_KEY << "=?";

            std::string sql(sqlStream.str());

            for (size_t i = 0; i < statements.size(); ++i) {
                statements[i].SetStatement(sql);
                Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
                attributes.push_back(
                    Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

                attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
                statements[i].SetParameters(attributes);
            }

            Aws::DynamoDB::Model::BatchExecuteStatementRequest request;
            request.SetStatements(statements);

            Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
            dynamoClient.BatchExecuteStatement(
                request);
            if (outcome.IsSuccess()) {
                const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
                outcome.GetResult();

                const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

                for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
                    const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

                    printMovieInfo(item);
                }
            }
            else {
                std::cerr << "Failed to retrieve the movies information: "
                         << outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        }

        std::cout << "Deleting the movie data with a batch \"DELETE\" statement."
                     << std::endl;

        // 6. Delete multiple movies using "Delete" statements. (BatchExecuteStatement)
        {
            Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
                titles.size());
            std::stringstream sqlStream;
            sqlStream << "DELETE FROM \\" << MOVIE_TABLE_NAME << "\\" WHERE "

```

```
<< TITLE_KEY << "=?" and " << YEAR_KEY << "=?";  
  
std::string sql(sqlStream.str());  
  
for (size_t i = 0; i < statements.size(); ++i) {  
    statements[i].SetStatement(sql);  
    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;  
    attributes.push_back(  
        Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));  
  
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));  
    statements[i].SetParameters(attributes);  
}  
  
Aws::DynamoDB::Model::BatchExecuteStatementRequest request;  
  
request.SetStatements(statements);  
  
Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =  
dynamoClient.BatchExecuteStatement(  
    request);  
  
if (!outcome.IsSuccess()) {  
    std::cerr << "Failed to delete the movies: "  
          << outcome.GetError().GetMessage() << std::endl;  
    return false;  
}  
}  
  
return true;  
}  
  
//! Create a DynamoDB table.  
/*!  
 \sa createDynamoDBTable()  
 \param tableName: The DynamoDB table's name.  
 \param clientConfiguration: Aws client configuration.  
 \return bool: Function succeeded.  
*/  
bool AwsDoc::DynamoDB::createDynamoDBTable(const Aws::String &tableName,  
                                             const Aws::Client::ClientConfiguration  
&clientConfiguration) {  
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);  
  
    bool movieTableAlreadyExisted = false;  
  
    {  
        Aws::DynamoDB::Model::CreateTableRequest request;  
  
        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;  
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);  
        yearAttributeDefinition.SetAttributeType(  
            Aws::DynamoDB::Model::ScalarAttributeType::N);  
        request.AddAttributeDefinitions(yearAttributeDefinition);  
  
        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;  
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);  
        yearAttributeDefinition.SetAttributeType(  
            Aws::DynamoDB::Model::ScalarAttributeType::S);  
        request.AddAttributeDefinitions(yearAttributeDefinition);  
  
        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;  
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(  
            Aws::DynamoDB::Model::KeyType::HASH);  
        request.AddKeySchema(yearKeySchema);  
    }
```

```
Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
    Aws::DynamoDB::Model::KeyType::RANGE);
request.AddKeySchema(yearKeySchema);

Aws::DynamoDB::Model::ProvisionedThroughput throughput;
throughput.WithReadCapacityUnits(
    PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
    PROVISIONED_THROUGHPUT_UNITS);
request.SetProvisionedThroughput(throughput);
request.SetTableName(MOVIE_TABLE_NAME);

std::cout << "Creating table " << MOVIE_TABLE_NAME << "..." << std::endl;
const Aws::DynamoDB::Model::CreateTableOutcome &result =
dynamoClient.CreateTable(
    request);
if (!result.IsSuccess()) {
    if (result.GetError().GetErrorCode() ==
        Aws::DynamoDBErrors::RESOURCE_IN_USE) {
        std::cout << "Table already exists." << std::endl;
        movieTableAlreadyExisted = true;
    }
    else {
        std::cerr << "Failed to create table: "
            << result.GetError().GetMessage();
        return false;
    }
}
// Wait for table to become active.
if (!movieTableAlreadyExisted) {
    std::cout << "Waiting for table " << MOVIE_TABLE_NAME
        << "' to become active...." << std::endl;
    if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
clientConfiguration)) {
        return false;
    }
    std::cout << "Table " << MOVIE_TABLE_NAME << "' created and active."
        << std::endl;
}
return true;
}

//! Delete a DynamoDB table.
/*!
\sa deleteDynamoTable()
\param tableName: The DynamoDB table's name.
\param clientConfiguration: Aws client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteDynamoTable(const Aws::String &tableName,
                                         const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
dynamoClient.DeleteTable(
        request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
            << result.GetResult().GetTableName()
```

```

        << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
            << std::endl;
    }
}

return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!!
 * \sa waitTableActive()
 * \param waitTableActive: The DynamoDB table's name.
 * \param clientConfiguration: Aws client configuration.
 * \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                         const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
            request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}

```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a struct that is a receiver for methods that can run PartiQL statements.

```
// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
// the
// specified table.
type PartiQLRunner struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// AddMovieBatch runs a batch of PartiQL INSERT statements to add multiple movies
// to the
// DynamoDB table.
func (runner PartiQLRunner) AddMovieBatch(movies []Movie) error {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
        movie.Info})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(fmt.Sprintf(
                "INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
                runner.TableName)),
            Parameters: params,
        }
    }

    _, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
    &dynamodb.BatchExecuteStatementInput{
        Statements: statementRequests,
    })
    if err != nil {
        log.Printf("Couldn't insert a batch of items with PartiQL. Here's why: %v\n",
        err)
    }
    return err
}

// GetMovieBatch runs a batch of PartiQL SELECT statements to get multiple movies
// from
// the DynamoDB table by title and year.
func (runner PartiQLRunner) GetMovieBatch(movies []Movie) ([]Movie, error) {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(
                fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
                runner.TableName)),
            Parameters: params,
        }
    }

    output, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
    &dynamodb.BatchExecuteStatementInput{
        Statements: statementRequests,
```

```
        })
    var outMovies []Movie
    if err != nil {
        log.Printf("Couldn't get a batch of items with PartiQL. Here's why: %v\n", err)
    } else {
        for _, response := range output.Responses {
            var movie Movie
            err = attributevalue.UnmarshalMap(response.Item, &movie)
            if err != nil {
                log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
            } else {
                outMovies = append(outMovies, movie)
            }
        }
    }
    return outMovies, err
}

// GetAllMovies runs a PartiQL SELECT statement to get all movies from the DynamoDB
// table.
// The results are projected to return only the title and rating of each movie.
func (runner PartiQLRunner) GetAllMovies() ([]map[string]interface{}, error) {
    var output []map[string]interface{}
    response, err := runner.DynamoDbClient.ExecuteStatement(context.TODO(),
        &dynamodb.ExecuteStatementInput{
            Statement: aws.String(
                fmt.Sprintf("SELECT title, info.rating FROM \"%v\"", runner.TableName)),
        })
    if err != nil {
        log.Printf("Couldn't get movies. Here's why: %v\n", err)
    } else {
        err = attributevalue.UnmarshalListOfMaps(response.Items, &output)
        if err != nil {
            log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
        }
    }
    return output, err
}

// UpdateMovieBatch runs a batch of PartiQL UPDATE statements to update the rating
// of
// multiple movies that already exist in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovieBatch(movies []Movie, ratings []float64)
    error {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{ratings[index],
            movie.Title, movie.Year})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(
                fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
                    runner.TableName)),
            Parameters: params,
        }
    }
    _, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
        &dynamodb.BatchExecuteStatementInput{
            Statements: statementRequests,
    })
    return err
}
```

```
        })
        if err != nil {
            log.Printf("Couldn't update the batch of movies. Here's why: %v\n", err)
        }
        return err
    }

    // DeleteMovieBatch runs a batch of PartiQL DELETE statements to remove multiple
    // movies
    // from the DynamoDB table.
    func (runner PartiQLRunner) DeleteMovieBatch(movies []Movie) error {
        statementRequests := make([]types.BatchStatementRequest, len(movies))
        for index, movie := range movies {
            params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
            if err != nil {
                panic(err)
            }
            statementRequests[index] = types.BatchStatementRequest{
                Statement: aws.String(
                    fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
                Parameters: params,
            }
        }

        _, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
        &dynamodb.BatchExecuteStatementInput{
            Statements: statementRequests,
        })
        if err != nil {
            log.Printf("Couldn't delete the batch of movies. Here's why: %v\n", err)
        }
        return err
    }
```

Run a scenario that creates a table and runs batches of PartiQL queries.

```
// RunPartiQLBatchScenario shows you how to use the AWS SDK for Go
// to run batches of PartiQL statements to query a table that stores data about
// movies.
//
// * Use batches of PartiQL statements to add, get, update, and delete data for
// individual movies.
//
// This example creates an Amazon DynamoDB service client from the specified
// sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// This example creates and deletes a DynamoDB table to use during the scenario.
func RunPartiQLBatchScenario(sdkConfig aws.Config, tableName string) {
    defer func() {
        if r := recover(); r != nil {
            fmt.Println("Something went wrong with the demo.")
        }
    }()
}

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon DynamoDB PartiQL batch demo.")
log.Println(strings.Repeat("-", 88))

tableBasics := actions.TableBasics{
```

```
DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
    TableName:      tableName,
}
runner := actions.PartiQLRunner{
    DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
    TableName:      tableName,
}

exists, err := tableBasics.TableExists()
if err != nil {
    panic(err)
}
if !exists {
    log.Printf("Creating table %v...\n", tableName)
    _, err = tableBasics.CreateMovieTable()
    if err != nil {
        panic(err)
    } else {
        log.Printf("Created table %v.\n", tableName)
    }
} else {
    log.Printf("Table %v already exists.\n", tableName)
}
log.Println(strings.Repeat("-", 88))

currentYear, _, _ := time.Now().Date()
customMovies := []actions.Movie{
    Title: "House PartiQL",
    Year:  currentYear - 5,
    Info: map[string]interface{}{
        "plot":   "Wacky high jinks result from querying a mysterious database.",
        "rating": 8.5},
    Title: "House PartiQL 2",
    Year:  currentYear - 3,
    Info: map[string]interface{}{
        "plot":   "Moderate high jinks result from querying another mysterious
database.",
        "rating": 6.5},
    Title: "House PartiQL 3",
    Year:  currentYear - 1,
    Info: map[string]interface{}{
        "plot":   "Tepid high jinks result from querying yet another mysterious
database.",
        "rating": 2.5},
},
}

log.Printf("Inserting a batch of movies into table '%v'.\n", tableName)
err = runner.AddMovieBatch(customMovies)
if err == nil {
    log.Printf("Added %v movies to the table.\n", len(customMovies))
}
log.Println(strings.Repeat("-", 88))

log.Println("Getting data for a batch of movies.")
movies, err := runner.GetMovieBatch(customMovies)
if err == nil {
    for _, movie := range movies {
        log.Println(movie)
    }
}
log.Println(strings.Repeat("-", 88))

newRatings := []float64{7.7, 4.4, 1.1}
log.Println("Updating a batch of movies with new ratings.")
err = runner.UpdateMovieBatch(customMovies, newRatings)
```

```
if err == nil {
    log.Printf("Updated %v movies with new ratings.\n", len(customMovies))
}
log.Println(strings.Repeat("-", 88))

log.Println("Getting projected data from the table to verify our update.")
projections, err := runner.GetAllMovies()
if err == nil {
    for _, projection := range projections {
        log.Println(projection)
    }
}
log.Println(strings.Repeat("-", 88))

log.Println("Deleting a batch of movies.")
err = runner.DeleteMovieBatch(customMovies)
if err == nil {
    log.Printf("Deleted %v movies.\n", len(customMovies))
}

err = tableBasics.DeleteTable()
if err == nil {
    log.Printf("Deleted table %v.\n", tableBasics.TableName)
}

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public class ScenarioPartiQLBatch {

    public static void main(String [] args) throws IOException {

        String tableName = "MoviesPartiQBatch";
        ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .credentialsProvider(credentialsProvider)
            .region(region)
            .build();

        System.out.println("***** Creating an Amazon DynamoDB table named
"+tableName +" with a key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("***** Adding multiple records into the "+ tableName
+" table using a batch command.");
        putRecordBatch(ddb);
```

```
        System.out.println("***** Updating multiple records using a batch
command.");
        updateTableItemBatch(ddb);

        System.out.println("***** Deleting multiple records using a batch
command.");
        deleteItemBatch(ddb);

        System.out.println("***** Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .provisionedThroughput(ProvisionedThroughput.builder()
                .readCapacityUnits(new Long(10))
                .writeCapacityUnits(new Long(10))
                .build())
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
            dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " +newTable + " was successfully created.");
        }
    }
}
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecordBatch(DynamoDbClient ddb) {
        String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?, 'title' : ?, 'info' : ?}";
        try {
            // Create three movies to add to the Amazon DynamoDB table.
            // Set data for Movie 1.
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("My Movie 1")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            BatchStatementRequest statementRequestMovie1 =
                BatchStatementRequest.builder()
                    .statement(sqlStatement)
                    .parameters(parameters)
                    .build();

            // Set data for Movie 2.
            List<AttributeValue> parametersMovie2 = new ArrayList<>();
            AttributeValue attMovie2 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

            AttributeValue attMovie2A = AttributeValue.builder()
                .s("My Movie 2")
                .build();

            AttributeValue attMovie2B = AttributeValue.builder()
                .s("No Information")
                .build();

            parametersMovie2.add(attMovie2);
            parametersMovie2.add(attMovie2A);
            parametersMovie2.add(attMovie2B);

            BatchStatementRequest statementRequestMovie2 =
                BatchStatementRequest.builder()
                    .statement(sqlStatement)
                    .parameters(parametersMovie2)
                    .build();

            // Set data for Movie 3.
            List<AttributeValue> parametersMovie3 = new ArrayList<>();
            AttributeValue attMovie3 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();
```

```
AttributeValue attMovie3A = AttributeValue.builder()  
    .s("My Movie 3")  
    .build();  
  
AttributeValue attMovie3B = AttributeValue.builder()  
    .s("No Information")  
    .build();  
  
parametersMovie3.add(attMovie3);  
parametersMovie3.add(attMovie3A);  
parametersMovie3.add(attMovie3B);  
  
BatchStatementRequest statementRequestMovie3 =  
BatchStatementRequest.builder()  
    .statement(sqlStatement)  
    .parameters(parametersMovie3)  
    .build();  
  
// Add all three movies to the list.  
List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();  
myBatchStatementList.add(statementRequestMovie1);  
myBatchStatementList.add(statementRequestMovie2);  
myBatchStatementList.add(statementRequestMovie3);  
  
BatchExecuteStatementRequest batchRequest =  
BatchExecuteStatementRequest.builder()  
    .statements(myBatchStatementList)  
    .build();  
  
BatchExecuteStatementResponse response =  
ddb.batchExecuteStatement(batchRequest);  
System.out.println("ExecuteStatement successful: "+  
response.toString());  
System.out.println("Added new movies using a batch command.");  
  
} catch (DynamoDbException e) {  
    System.err.println(e.getMessage());  
    System.exit(1);  
}  
}  
  
public static void updateTableItemBatch(DynamoDbClient ddb){  
    String sqlStatement = "UPDATE MoviesPartiQBatch SET info = 'directors\':  
\\\"Merian C. Cooper\\\",\\\"Ernest B. Schoedsack\\' where year=? and title=?";  
    List<AttributeValue> parametersRec1 = new ArrayList<>();  
  
    // Update three records.  
    AttributeValue att1 = AttributeValue.builder()  
        .n(String.valueOf("2022"))  
        .build();  
  
    AttributeValue att2 = AttributeValue.builder()  
        .s("My Movie 1")  
        .build();  
  
    parametersRec1.add(att1);  
    parametersRec1.add(att2);  
  
    BatchStatementRequest statementRequestRec1 =  
BatchStatementRequest.builder()  
    .statement(sqlStatement)  
    .parameters(parametersRec1)  
    .build();  
  
    // Update record 2.  
    List<AttributeValue> parametersRec2 = new ArrayList<>();
```

```
AttributeValue attRec2 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec2a = AttributeValue.builder()
    .s("My Movie 2")
    .build();

parametersRec2.add(attRec2);
parametersRec2.add(attRec2a);
BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec2)
    .build();

// Update record 3.
List<AttributeValue> parametersRec3 = new ArrayList<>();
AttributeValue attRec3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec3a = AttributeValue.builder()
    .s("My Movie 3")
    .build();

parametersRec3.add(attRec3);
parametersRec3.add(attRec3a);
BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec3)
    .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
myBatchStatementList.add(statementRequestRec1);
myBatchStatementList.add(statementRequestRec2);
myBatchStatementList.add(statementRequestRec3);

BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
    .statements(myBatchStatementList)
    .build();

try {
    BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
    System.out.println("ExecuteStatement successful: "+
response.toString());
    System.out.println("Updated three movies using a batch command.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("Item was updated!");
}

public static void deleteItemBatch(DynamoDbClient ddb){
    String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and
title=?";
    List<AttributeValue> parametersRec1 = new ArrayList<>();

    // Specify three records to delete.
    AttributeValue att1 = AttributeValue.builder()
```

```
.n(String.valueOf("2022"))
.build();

AttributeValue att2 = AttributeValue.builder()
.s("My Movie 1")
.build();

parametersRec1.add(att1);
parametersRec1.add(att2);

BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
.statement(sqlStatement)
.parameters(parametersRec1)
.build();

// Specify record 2.
List<AttributeValue> parametersRec2 = new ArrayList<>();
AttributeValue attRec2 = AttributeValue.builder()
.n(String.valueOf("2022"))
.build();

AttributeValue attRec2a = AttributeValue.builder()
.s("My Movie 2")
.build();

parametersRec2.add(attRec2);
parametersRec2.add(attRec2a);
BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
.statement(sqlStatement)
.parameters(parametersRec2)
.build();

// Specify record 3.
List<AttributeValue> parametersRec3 = new ArrayList<>();
AttributeValue attRec3 = AttributeValue.builder()
.n(String.valueOf("2022"))
.build();

AttributeValue attRec3a = AttributeValue.builder()
.s("My Movie 3")
.build();

parametersRec3.add(attRec3);
parametersRec3.add(attRec3a);

BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
.statement(sqlStatement)
.parameters(parametersRec3)
.build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
myBatchStatementList.add(statementRequestRec1);
myBatchStatementList.add(statementRequestRec2);
myBatchStatementList.add(statementRequestRec3);

BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
.statements(myBatchStatementList)
.build();

try {
ddb.batchExecuteStatement(batchRequest);
```

```
        System.out.println("Deleted three movies using a batch command.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}

private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
ddb, String statement, List<AttributeValue> parameters ) {
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .statement(statement)
        .parameters(parameters)
        .build();

    return ddb.executeStatement(request);
}
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
export const REGION = "eu-west-1"; // For example, "us-east-1".
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: REGION });
```

Create the document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
```

```
// Whether to automatically convert empty strings, blobs, and sets to `null`.
convertEmptyValues: false, // false, by default.
// Whether to remove undefined values while marshalling.
removeUndefinedValues: false, // false, by default.
// Whether to convert typeof object to map attribute.
convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

const translateConfig = { marshallOptions, unmarshallOptions };

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, translateConfig);

export { ddbDocClient };
```

Query items by batch.

```
/*
import fs from "fs";
// A practical functional library used to split the data into segments.
import * as R from "ramda";
import { ddbClient } from "../libs/ddbClient.js";
import { ddbDocClient } from "../libs/ddbDocClient.js";
import { BatchWriteCommand } from "@aws-sdk/lib-dynamodb";
import {
    CreateTableCommand,
    BatchExecuteStatementCommand,
} from "@aws-sdk/client-dynamodb";
if (process.argv.length < 6) {
    console.log(
        "Usage: node partiQL_basics.js <tableName> <movieTitle1> <movieYear1>
<movieTitle1> <movieYear1> <producer1> <producer2> \n" +
        "Example: node partiQL_basics.js Movies_batch 2006 'The Departed' 2013 '2
Guns' 'New View Films' 'Old Thyme Films'"
    );
}

const tableName = process.argv[2];
const movieYear1 = parseInt(process.argv[3]);
const movieTitle1 = process.argv[4];
const movieYear2 = parseInt(process.argv[5]);
const movieTitle2 = process.argv[6];
const producer1 = process.argv[7];
const producer2 = process.argv[8];

// Helper function to delay running the code while the AWS service calls wait for
// responses.
function wait(ms) {
    var start = Date.now();
    var end = start;
    while (end < start + ms) {
        end = Date.now();
    }
}
// Set the parameters.

export const run = async (
    tableName,
```

```
    movieYear1,
    movieTitle1,
    movieYear2,
    movieTitle2,
    producer1,
    producer2
) => {
  try {
    console.log("Creating table ...");
    // Set the parameters.
    const params = {
      AttributeDefinitions: [
        {
          AttributeName: "title",
          AttributeType: "S",
        },
        {
          AttributeName: "year",
          AttributeType: "N",
        },
      ],
      KeySchema: [
        {
          AttributeName: "title",
          KeyType: "HASH",
        },
        {
          AttributeName: "year",
          KeyType: "RANGE",
        },
      ],
      ProvisionedThroughput: {
        ReadCapacityUnits: 5,
        WriteCapacityUnits: 5,
      },
      TableName: tableName,
    };
    const data = await ddbClient.send(new CreateTableCommand(params));
    console.log("Waiting for table to be created...");
    wait(10000);
    console.log(
      "Table created. Table name is ",
      data.TableDescription.TableName
    );
    try {
      // Before you run this example, download 'movies.json' from https://
      // docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Js.02.html,
      // and put it in the same folder as the example.
      // Get the movie data parse to convert into a JSON object.
      const allMovies = JSON.parse(fs.readFileSync("moviedata.json", "utf8"));
      // Split the table into segments of 25.
      const dataSegments = R.splitEvery(25, allMovies);
      // Loop batch write operation 10 times to upload 250 items.
      console.log("Writing movies in batch to table... ");
      for (let i = 0; i < 10; i++) {
        const segment = dataSegments[i];
        for (let j = 0; j < 25; j++) {
          const params = {
            RequestItems: {
              [tableName]: [
                {
                  // Destination Amazon DynamoDB table name.
                  PutRequest: {
                    Item: {
                      year: segment[j].year,
                      title: segment[j].title,

```

```
                info: segment[j].info,
            },
        ],
    ],
},
ddbDocClient.send(new BatchWriteCommand(params));
}
}
wait(10000);
console.log("Success, movies written to table.");
try {
    console.log("Getting movie....");
    const params = {
        Statements: [
            {
                Statement:
                    "SELECT * FROM " + tableName + " where title=? and year=?",
                Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
            },
            {
                Statement:
                    "SELECT * FROM " + tableName + " where title=? and year=?",
                Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
            },
        ],
    };
    const data = await ddbDocClient.send(
        new BatchExecuteStatementCommand(params)
    );
    console.log("Success. The query return the following data.", data);
    for (let i = 0; i < data.Responses.length; i++) {
        console.log(data.Responses[i].Item.year);
        console.log(data.Responses[i].Item.title);
    }
    try {
        const params = {
            Statements: [
                {
                    Statement:
                        "DELETE FROM " + tableName + " where title=? and year=?",
                    Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
                },
                {
                    Statement:
                        "DELETE FROM " + tableName + " where title=? and year=?",
                    Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
                },
            ],
        };
        await ddbDocClient.send(
            new BatchExecuteStatementCommand(params)
        );
        console.log("Success. Items deleted by batch.");
        try {
            const params = {
                Statements: [
                    {
                        Statement:
                            "INSERT INTO " +
                            tableName +
                            " value {'title':?, 'year':?}",
                        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
                    },
                    {

```

```
Statement:  
    "INSERT INTO " +  
    tableName +  
    " value {'title':?, 'year':?}"  
Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],  
    ],  
    ],  
};  
await ddbDocClient.send(  
    new BatchExecuteStatementCommand(params)  
);  
console.log("Success. Items added by batch.");  
try {  
    const params = {  
        Statements: [  
            {  
                Statement:  
                    "UPDATE " +  
                    tableName +  
                    " SET Producer=? where title=? and year=?"  
                Parameters: [  
                    { S: producer1 },  
                    { S: movieTitle1 },  
                    { N: movieYear1 },  
                ],  
            },  
            {  
                Statement:  
                    "UPDATE " +  
                    tableName +  
                    " SET Producer=? where title=? and year=?"  
                Parameters: [  
                    { S: producer2 },  
                    { S: movieTitle2 },  
                    { N: movieYear2 },  
                ],  
            },  
            ],  
        ],  
    };  
    console.log("Updating movies...");  
    await ddbDocClient.send(  
        new BatchExecuteStatementCommand(params)  
    );  
    console.log("Success. Items updated by batch.");  
    return "Run successfully"; // For unit tests.  
} catch (err) {  
    console.log("Error updating items by batch. ", err);  
}  
} catch (err) {  
    console.log("Error adding items to table by batch. ", err);  
}  
} catch (err) {  
    console.log("Error deleting movies by batch. ", err);  
}  
} catch (err) {  
    console.log("Error getting movies by batch. ", err);  
}  
} catch (err) {  
    console.log("Error adding movies by batch. ", err);  
}  
} catch (err) {  
    console.log("Error creating table. ", err);  
}  
};  
run(  
    tableName,
```

```
    movieYear1,
    movieTitle1,
    movieYear2,
    movieTitle2,
    producer1,
    producer2
);
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun main() {

    val ddb = DynamoDbClient { region = "us-east-1" }
    val tableName = "MoviesPartiQLBatch"
    println("Creating an Amazon DynamoDB table named $tableName with a key named id and a sort key named title.")
    createTablePartiQLBatch(ddb, tableName, "year")
    putRecordBatch(ddb)
    updateTableItemBatchBatch(ddb)
    deleteItemsBatch(ddb)
    deleteTablePartiQLBatch(tableName)
}

suspend fun createTablePartiQLBatch(ddb: DynamoDbClient, tableNameVal: String, key: String) {

    val attDef = AttributeDefinition {
        attributeName = key
        attributeType = ScalarAttributeType.N
    }

    val attDef1 = AttributeDefinition {
        attributeName = "title"
        attributeType = ScalarAttributeType.S
    }

    val keySchemaVal = KeySchemaElement {
        attributeName = key
        keyType = KeyType.Hash
    }

    val keySchemaVal1 = KeySchemaElement {
        attributeName = "title"
        keyType = KeyType.Range
    }

    val provisionedVal = ProvisionedThroughput {
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }
```

```
val request = CreateTableRequest {  
    attributeDefinitions = listOf(attDef, attDef1)  
    keySchema = listOf(keySchemaVal, keySchemaVal1)  
    provisionedThroughput = provisionedVal  
    tableName = tableNameVal  
}  
  
val response = ddb.createTable(request)  
ddb.waitUntilTableExists { // suspend call  
    tableName = tableNameVal  
}  
println("The table was successfully created  
${response.tableDescription?.tableArn}")  
}  
  
suspend fun putRecordBatch(ddb: DynamoDbClient) {  
  
    val sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?, 'title' : ?,  
    'info' : ?}"  
  
    // Create three movies to add to the Amazon DynamoDB table.  
    val parametersMovie1 = mutableListOf<AttributeValue>()  
    parametersMovie1.add(AttributeValue.N("2022"))  
    parametersMovie1.add(AttributeValue.S("My Movie 1"))  
    parametersMovie1.add(AttributeValue.S("No Information"))  
  
    val statementRequestMovie1 = BatchStatementRequest {  
        statement = sqlStatement  
        parameters = parametersMovie1  
    }  
  
    // Set data for Movie 2.  
    val parametersMovie2 = mutableListOf<AttributeValue>()  
    parametersMovie2.add(AttributeValue.N("2022"))  
    parametersMovie2.add(AttributeValue.S("My Movie 2"))  
    parametersMovie2.add(AttributeValue.S("No Information"))  
  
    val statementRequestMovie2 = BatchStatementRequest {  
        statement = sqlStatement  
        parameters = parametersMovie2  
    }  
  
    // Set data for Movie 3.  
    val parametersMovie3 = mutableListOf<AttributeValue>()  
    parametersMovie3.add(AttributeValue.N("2022"))  
    parametersMovie3.add(AttributeValue.S("My Movie 3"))  
    parametersMovie3.add(AttributeValue.S("No Information"))  
  
    val statementRequestMovie3 = BatchStatementRequest {  
        statement = sqlStatement  
        parameters = parametersMovie3  
    }  
  
    // Add all three movies to the list.  
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()  
    myBatchStatementList.add(statementRequestMovie1)  
    myBatchStatementList.add(statementRequestMovie2)  
    myBatchStatementList.add(statementRequestMovie3)  
  
    val batchRequest = BatchExecuteStatementRequest {  
        statements = myBatchStatementList  
    }  
    val response = ddb.batchExecuteStatement(batchRequest)  
    println("ExecuteStatement successful: " + response.toString())  
    println("Added new movies using a batch command.")
```

```

}

suspend fun updateTableItemBatchBatch(ddb: DynamoDbClient) {
    val sqlStatement =
        "UPDATE MoviesPartiQBatch SET info = 'directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack\" where year=? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))
    val statementRequestRec1 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec1
    }

    // Update record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec2
    }

    // Update record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec3
    }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest = BatchExecuteStatementRequest {
        statements = myBatchStatementList
    }

    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: $response")
    println("Updated three movies using a batch command.")
    println("Items were updated!")
}

suspend fun deleteItemsBatch(ddb: DynamoDbClient) {

    // Specify three records to delete.
    val sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and title=?"
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))

    val statementRequestRec1 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec1
    }

    // Specify record 2.
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 = BatchStatementRequest {
}

```

```
        statement = sqlStatement
        parameters = parametersRec2
    }

    // Specify record 3.
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec3
    }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)

    val batchRequest = BatchExecuteStatementRequest {
        statements = myBatchStatementList
    }

    ddb.batchExecuteStatement(batchRequest)
    println("Deleted three movies using a batch command.")
}

suspend fun deleteTablePartiQLBatch(tableNameVal: String) {

    val request = DeleteTableRequest {
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\AttributeValue;

class GettingStartedWithPartiQLBatch
{
    public function run()
    {
        echo("-----\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using
PHP!\n");
```

```
echo("-----\n");

$uuid = uniqid();
$service = new DynamoDb\DynamoDBService();

$tableName = "partiql_demo_table_{$uuid}";
$service->createTable(
    $tableName,
    [
        new DynamoDBAttribute('year', 'N', 'HASH'),
        new DynamoDBAttribute('title', 'S', 'RANGE')
    ]
);

echo "Waiting for table...";
$service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
$tableName]);
echo "table $tableName found!\n";

echo "What's the name of the last movie you watched?\n";
while (empty($movieName)) {
    $movieName = testable_readline("Movie name: ");
}
echo "And what year was it released?\n";
$movieYear = "year";
while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
    $movieYear = testable_readline("Year released: ");
}
$key = [
    'Item' => [
        'year' => [
            'N' => "$movieYear",
        ],
        'title' => [
            'S' => $movieName,
        ],
    ],
];
list($statement, $parameters) = $service-
>buildStatementAndParameters("INSERT", $tableName, $key);
$service->insertItemByPartiQLBatch($statement, $parameters);

echo "How would you rate the movie from 1-10?\n";
$rating = 0;
while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
    $rating = testable_readline("Rating (1-10): ");
}
echo "What was the movie about?\n";
while (empty($plot)) {
    $plot = testable_readline("Plot summary: ");
}
$attributes = [
    new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
    new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
];

list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
$service->updateItemByPartiQLBatch($statement, $parameters);
echo "Movie added and updated.\n";

$batch = json_decode(loadMovieData());
$service->writeBatch($tableName, $batch);
```

```

$movie = $service->getItemByPartiQLBatch($tableName, [$key]);
echo "\nThe movie {$movie['Responses'][0]['Item']['title']]['S']"
was released in {$movie['Responses'][0]['Item']['year']]['N'].\n";
echo "What rating would you like to give {$movie['Responses'][0]['Item']
['title']]['S']}?\n";
$rating = 0;
while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 || 
$rating > 10) {
    $rating = testable_readline("Rating (1-10): ");
}
$attributes = [
    new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
    new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
];
list($statement, $parameters) = $service-
>buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
$service->updateItemByPartiQLBatch($statement, $parameters);

$movie = $service->getItemByPartiQLBatch($tableName, [$key]);
echo "Okay, you have rated {$movie['Responses'][0]['Item']['title']]['S']"
as a {$movie['Responses'][0]['Item']['rating']]['N']\n";

$service->deleteItemByPartiQLBatch($statement, $parameters);
echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

echo "That's okay though. The book was better. Now, for something lighter,
in what year were you born?\n";
$birthYear = "not a number";
while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
    $birthYear = testable_readline("Birth year: ");
}
$birthKey = [
    'Key' => [
        'year' => [
            'N' => "$birthYear",
        ],
    ],
];
$result = $service->query($tableName, $birthKey);
$marshal = new Marshaler();
echo "Here are the movies in our collection released the year you were
born:\n";
$oops = "Oops! There were no movies released in that year (that we know
of).\n";
$display = "";
foreach ($result['Items'] as $movie) {
    $movie = $marshal->unmarshalItem($movie);
    $display .= $movie['title'] . "\n";
}
echo ($display) ?: $oops;

$yearsKey = [
    'Key' => [
        'year' => [
            'N' => [
                'minRange' => 1990,
                'maxRange' => 1999,
            ],
        ],
    ],
];
$filter = "year between 1990 and 1999";
echo "\nHere's a list of all the movies released in the 90s:\n";
$result = $service->scan($tableName, $yearsKey, $filter);
foreach ($result['Items'] as $movie) {

```

```

        $movie = $marshal->unmarshalItem($movie);
        echo $movie['title'] . "\n";
    }

    echo "\nCleaning up this demo by deleting table $tableName...\n";
    $service->deleteTable($tableName);
}
}

public function insertItemByPartiQLBatch(string $statement, array $parameters)
{
    $this->dynamoDbClient->batchExecuteStatement([
        'Statements' => [
            [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ],
        ],
    ]);
}

public function getItemByPartiQLBatch(string $tableName, array $keys): Result
{
    $statements = [];
    foreach ($keys as $key) {
        list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
        $statements[] = [
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ];
    }

    return $this->dynamoDbClient->batchExecuteStatement([
        'Statements' => $statements,
    ]);
}

public function updateItemByPartiQLBatch(string $statement, array $parameters)
{
    $this->dynamoDbClient->batchExecuteStatement([
        'Statements' => [
            [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ],
        ],
    ]);
}

public function deleteItemByPartiQLBatch(string $statement, array $parameters)
{
    $this->dynamoDbClient->batchExecuteStatement([
        'Statements' => [
            [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ],
        ],
    ]);
}

```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that can run batches of PartiQL statements.

```
from datetime import datetime
from decimal import Decimal
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

from scaffold import Scaffold

logger = logging.getLogger(__name__)

class PartiQLBatchWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource

    def run_partiql(self, statements, param_list):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because the
        resource transforms input and output from plain old Python objects (POPOs)
        to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statements: The batch of PartiQL statements.
        :param param_list: The batch of PartiQL parameters that are associated with
                           each statement. This list must be in the same order as
                           the
                           statements.
        :return: The responses returned from running the statements, if any.
        """
        try:
            output = self.dyn_resource.meta.client.batch_execute_statement(
                Statements=[{
                    'Statement': statement, 'Parameters': params
                } for statement, params in zip(statements, param_list)])
        except ClientError as err:
            if err.response['Error']['Code'] == 'ResourceNotFoundException':
                logger.error(
                    "Couldn't execute batch of PartiQL statements because the table"
                    "does not exist.")
            else:
                logger.error(
                    "Couldn't execute batch of PartiQL statements. Here's why: %s",
                    err.response['Error']['Code'], err.response['Error']
                    ['Message'])

```

```
    raise
else:
    return output
```

Run a scenario that creates a table and runs PartiQL queries in batches.

```
def run_scenario(scaffold, wrapper, table_name):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the Amazon DynamoDB PartiQL batch statement demo.")
    print('*'*88)

    print(f"Creating table '{table_name}' for the demo...")
    scaffold.create_table(table_name)
    print('*'*88)

    movie_data = [
        {
            'title': f"House PartiQL",
            'year': datetime.now().year - 5,
            'info': {
                'plot': "Wacky high jinks result from querying a mysterious database.",
                'rating': Decimal('8.5')}},
        {
            'title': f"House PartiQL 2",
            'year': datetime.now().year - 3,
            'info': {
                'plot': "Moderate high jinks result from querying another mysterious database.",
                'rating': Decimal('6.5')}},
        {
            'title': f"House PartiQL 3",
            'year': datetime.now().year - 1,
            'info': {
                'plot': "Tepid high jinks result from querying yet another mysterious database.",
                'rating': Decimal('2.5')}},
    ]

    print(f"Inserting a batch of movies into table '{table_name}.")
    statements = [
        f"INSERT INTO \\"{table_name}\\" "
        f"VALUES {{'title': ?, 'year': ?, 'info': ?}}] * len(movie_data)
    params = [list(movie.values()) for movie in movie_data]
    wrapper.run_partiql(statements, params)
    print("Success!")
    print('*'*88)

    print(f"Getting data for a batch of movies.")
    statements = [
        f"SELECT * FROM \\"{table_name}\\" WHERE title=? AND year=?"] *
    len(movie_data)
    params = [[movie['title'], movie['year']] for movie in movie_data]
    output = wrapper.run_partiql(statements, params)
    for item in output['Responses']:
        print(f"\n{item['Item']['title']}, {item['Item']['year']}")
        pprint(item['Item'])
    print('*'*88)

    ratings = [Decimal('7.7'), Decimal('5.5'), Decimal('1.3')]
    print(f"Updating a batch of movies with new ratings.")
    statements = [
        f"UPDATE \\"{table_name}\\" SET info.rating=? "
        f"WHERE title=? AND year=?"] * len(movie_data)
    params = [
        [rating, movie['title'], movie['year']] for rating, movie in zip(ratings,
    movie_data)]
```

```
wrapper.run_partiql(statements, params)
print("Success!")
print('*'*88)

print(f"Getting projected data from the table to verify our update.")
output = wrapper.dyn_resource.meta.client.execute_statement(
    Statement=f'SELECT title, info.rating FROM "{table_name}"')
pprint(output['Items'])
print('*'*88)

print(f"Deleting a batch of movies from the table.")
statements = [
    f'DELETE FROM \'{table_name}\' WHERE title=? AND year=?'] * len(movie_data)
params = [[movie['title'], movie['year']] for movie in movie_data]
wrapper.run_partiql(statements, params)
print("Success!")
print('*'*88)

print(f"Deleting table '{table_name}'...")
scaffold.delete_table()
print('*'*88)

print("\nThanks for watching!")
print('*'*88)

if __name__ == '__main__':
    try:
        dyn_res = boto3.resource('dynamodb')
        scaffold = Scaffold(dyn_res)
        movies = PartiQLBatchWrapper(dyn_res)
        run_scenario(scaffold, movies, 'doc-example-table-partiql-movies')
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Python (Boto3) API Reference*.

## Query a DynamoDB table using PartiQL and an AWS SDK

The following code examples show how to:

- Get an item by running a SELECT statement.
- Add an item by running an INSERT statement.
- Update an item by running an UPDATE statement.
- Delete an item by running a DELETE statement.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace PartiQL_Basics_Scenario
{
    public class PartiQLMethods
    {
```

```
private static readonly AmazonDynamoDBClient Client = new
AmazonDynamoDBClient();

/// <summary>
/// Inserts movies imported from a JSON file into the movie table by
/// using an Amazon DynamoDB PartiQL INSERT statement.
/// </summary>
/// <param name="tableName">The name of the table where the movie
/// information will be inserted.</param>
/// <param name="movieFileName">The name of the JSON file that contains
/// movie information.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the insert operation.</returns>
public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
{
    // Get the list of movies from the JSON file.
    var movies = ImportMovies(movieFileName);

    var success = false;

    if (movies is not null)
    {
        // Insert the movies in a batch using PartiQL. Because the
        // batch can contain a maximum of 25 items, insert 25 movies
        // at a time.
        string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,"
'year': ?}}";
        var statements = new List<BatchStatementRequest>();

        try
        {
            for (var indexOffset = 0; indexOffset < 250; indexOffset += 25)
            {
                for (var i = indexOffset; i < indexOffset + 25; i++)
                {
                    statements.Add(new BatchStatementRequest
                    {
                        Statement = insertBatch,
                        Parameters = new List<AttributeValue>
                        {
                            new AttributeValue { S = movies[i].Title },
                            new AttributeValue { N =
movies[i].Year.ToString() },
                        },
                    });
                }

                var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
{
    Statements = statements,
});
                // Wait between batches for movies to be successfully
added.
                System.Threading.Thread.Sleep(3000);

                success = response.StatusCode ==
System.Net.HttpStatusCode.OK;
                // Clear the list of statements for the next batch.
                statements.Clear();
            }
        }
    }
}
```

```

        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine(ex.Message);
        }
    }

    return success;
}

/// <summary>
/// Loads the contents of a JSON file into a list of movies to be
/// added to the DynamoDB table.
/// </summary>
/// <param name="movieFileName">The full path to the JSON file.</param>
/// <returns>A generic list of movie objects.</returns>
public static List<Movie> ImportMovies(string movieFileName)
{
    if (!File.Exists(movieFileName))
    {
        return null;
    }

    using var sr = new StreamReader(movieFileName);
    string json = sr.ReadToEnd();
    var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

    if (allMovies is not null)
    {
        // Return the first 250 entries.
        return allMovies.GetRange(0, 250);
    }
    else
    {
        return null;
    }
}

/// <summary>
/// Uses a PartiQL SELECT statement to retrieve a single movie from the
/// movie database.
/// </summary>
/// <param name="tableName">The name of the movie table.</param>
/// <param name="movieTitle">The title of the movie to retrieve.</param>
/// <returns>A list of movie data. If no movie matches the supplied
/// title, the list is empty.</returns>
public static async Task<List<Dictionary<string, AttributeValue>>>
GetSingleMovie(string tableName, string movieTitle)
{
    string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
    var parameters = new List<AttributeValue>
    {
        new AttributeValue { S = movieTitle },
    };

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = selectSingle,
    Parameters = parameters,
});
    return response.Items;
}

```

```

    ///<summary>
    /// Retrieve multiple movies by year using a SELECT statement.
    ///</summary>
    ///<param name="tableName">The name of the movie table.</param>
    ///<param name="year">The year the movies were released.</param>
    ///<returns></returns>
    public static async Task<List<Dictionary<string, AttributeValue>>>
GetMovies(string tableName, int year)
{
    string selectSingle = $"SELECT * FROM {tableName} WHERE year = ?";
    var parameters = new List<AttributeValue>
    {
        new AttributeValue { N = year.ToString() },
    };

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = selectSingle,
    Parameters = parameters,
});
    return response.Items;
}

    ///<summary>
    /// Inserts a single movie into the movies table.
    ///</summary>
    ///<param name="tableName">The name of the table.</param>
    ///<param name="movieTitle">The title of the movie to insert.</param>
    ///<param name="year">The year that the movie was released.</param>
    ///<returns>A Boolean value that indicates the success or failure of
    /// the INSERT operation.</returns>
    public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
{
    string insertBatch = $"INSERT INTO {tableName} VALUE {'title': ?, 'year': ?}";
    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = insertBatch,
    Parameters = new List<AttributeValue>
    {
        new AttributeValue { S = movieTitle },
        new AttributeValue { N = year.ToString() },
    },
});
    return response.StatusCode == System.Net.HttpStatusCode.OK;
}

    ///<summary>
    /// Updates a single movie in the table, adding information for the
    /// producer.
    ///</summary>
    ///<param name="tableName">the name of the table.</param>
    ///<param name="producer">The name of the producer.</param>
    ///<param name="movieTitle">The movie title.</param>
    ///<param name="year">The year the movie was released.</param>

```

```

    ///<returns>A Boolean value that indicates the success of the
    /// UPDATE operation.</returns>
    public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
{
    string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = insertSingle,
    Parameters = new List<AttributeValue>
{
    new AttributeValue { S = producer },
    new AttributeValue { S = movieTitle },
    new AttributeValue { N = year.ToString() },
},
});
}

return response.StatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Deletes a single movie from the table.
/// </summary>
/// <param name="tableName">The name of the table.</param>
/// <param name="movieTitle">The title of the movie to delete.</param>
/// <param name="year">The year that the movie was released.</param>
/// <returns>A Boolean value that indicates the success of the
/// DELETE operation.</returns>
public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
{
    var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = deleteSingle,
    Parameters = new List<AttributeValue>
{
    new AttributeValue { S = movieTitle },
    new AttributeValue { N = year.ToString() },
},
});
}

return response.StatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Displays the list of movies returned from a database query.
/// </summary>
/// <param name="items">The list of movie information to display.</param>
private static void DisplayMovies(List<Dictionary<string, AttributeValue>>
items)
{
    if (items.Count > 0)
    {
        Console.WriteLine($"Found {items.Count} movies.");
        items.ForEach(item =>
Console.WriteLine($"{item["year"].N}\t{item["title"].S}"));
    }
}

```

```

        }
    else
    {
        Console.WriteLine($"Didn't find a movie that matched the supplied
criteria.");
    }
}

}

/// <summary>
/// Uses a PartiQL SELECT statement to retrieve a single movie from the
/// movie database.
/// </summary>
/// <param name="tableName">The name of the movie table.</param>
/// <param name="movieTitle">The title of the movie to retrieve.</param>
/// <returns>A list of movie data. If no movie matches the supplied
/// title, the list is empty.</returns>
public static async Task<List<Dictionary<string, AttributeValue>>>
GetSingleMovie(string tableName, string movieTitle)
{
    string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
    var parameters = new List<AttributeValue>
    {
        new AttributeValue { S = movieTitle },
    };

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = selectSingle,
    Parameters = parameters,
});

    return response.Items;
}

/// <summary>
/// Inserts a single movie into the movies table.
/// </summary>
/// <param name="tableName">The name of the table.</param>
/// <param name="movieTitle">The title of the movie to insert.</param>
/// <param name="year">The year that the movie was released.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the INSERT operation.</returns>
public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
{
    string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?, 'year': ?}}";

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = insertBatch,
    Parameters = new List<AttributeValue>
    {
        new AttributeValue { S = movieTitle },
        new AttributeValue { N = year.ToString() },
    },
}

```

```
        });

        return response.StatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Updates a single movie in the table, adding information for the
    /// producer.
    /// </summary>
    /// <param name="tableName">the name of the table.</param>
    /// <param name="producer">The name of the producer.</param>
    /// <param name="movieTitle">The movie title.</param>
    /// <param name="year">The year the movie was released.</param>
    /// <returns>A Boolean value that indicates the success of the
    /// UPDATE operation.</returns>
    public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
{
    string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title
= ? AND year = ?";

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = insertSingle,
    Parameters = new List<AttributeValue>
    {
        new AttributeValue { S = producer },
        new AttributeValue { S = movieTitle },
        new AttributeValue { N = year.ToString() },
    },
});

    return response.StatusCode == System.Net.HttpStatusCode.OK;
}

    /// <summary>
    /// Deletes a single movie from the table.
    /// </summary>
    /// <param name="tableName">The name of the table.</param>
    /// <param name="movieTitle">The title of the movie to delete.</param>
    /// <param name="year">The year that the movie was released.</param>
    /// <returns>A Boolean value that indicates the success of the
    /// DELETE operation.</returns>
    public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
{
    var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = deleteSingle,
    Parameters = new List<AttributeValue>
    {
        new AttributeValue { S = movieTitle },
        new AttributeValue { N = year.ToString() },
    },
});

    return response.StatusCode == System.Net.HttpStatusCode.OK;
}
```

```
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// 1. Create a table. (CreateTable)
if
(AwsDoc::DynamoDB::createDynamoDBTable(AwsDoc::DynamoDB::MOVIE_TABLE_NAME,
                                         clientConfig)) {

    AwsDoc::DynamoDB::partiqlExecuteScenario(clientConfig);

    // 7. Delete the table. (DeleteTable)
    AwsDoc::DynamoDB::deleteDynamoTable(AwsDoc::DynamoDB::MOVIE_TABLE_NAME,
                                         clientConfig);
}

//! Scenario to modify and query a DynamoDB table using single PartiQL statements.
/*!
 \sa partiqlExecuteScenario()
 \param clientConfiguration: Aws client configuration.
 \return bool: Function succeeded.
 */
bool
AwsDoc::DynamoDB::partiqlExecuteScenario(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
            "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate
it? ",
                                         1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {\""
            << TITLE_KEY << "\": ?, \"" << YEAR_KEY << "\": ?, \""
            << INFO_KEY << "\": ?\"}";
    }

    request.SetStatement(sqlStream.str());

    // Create the parameter attributes.
    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
```

```

Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
    ALLOCATION_TAG.c_str());
ratingAttribute->SetN(rating);
infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
    ALLOCATION_TAG.c_str());
plotAttribute->SetS(plot);
infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
attributes.push_back(infoMapAttribute);
request.SetParameters(attributes);

Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to add a movie: " <<
outcome.GetError().GetMessage()
        << std::endl;
    return false;
}
}

std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
    << std::endl;

// 3. Get the data for the movie using a "Select" statement.
(ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM \\" << MOVIE_TABLE_NAME << "\\" WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
        request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to retrieve movie information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    else {
        // Print the retrieved movie information.
        const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

        if (items.size() == 1) {
            printMovieInfo(items[0]);
        }
    }
}

```

```

        }
    else {
        std::cerr << "Error: " << items.size() << " movies were retrieved.
"
                << " There should be only one movie." << std::endl;
    }
}

// 4. Update the data for the movie using an "Update" statement.
(ExecuteStatement)
{
    rating = askQuestionForFloatRange(
        Aws::String("\nLet's update your movie.\nYou rated it  ") +
        std::to_string(rating)
        + ", what new rating would you give it? ", 1, 10);

    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "UPDATE \\" << MOVIE_TABLE_NAME << "\\ SET "
        << INFO_KEY << "." << RATING_KEY << "=? WHERE "
        << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to update a movie: "
            << outcome.GetError().GetMessage();
        return false;
    }
}

std::cout << "\nUpdated " << title << "' with new attributes:" << std::endl;

// 5. Get the updated data for the movie using a "Select" statement.
(ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM \\" << MOVIE_TABLE_NAME << "\\ WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to retrieve the movie information: "
            << outcome.GetError().GetMessage() << std::endl;
    }
}

```

```

        return false;
    }
    else {
        const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

        if (items.size() == 1) {
            printMovieInfo(items[0]);
        }
        else {
            std::cerr << "Error: " << items.size() << " movies were retrieved.
"
            << " There should be only one movie." << std::endl;
        }
    }
}

std::cout << "Deleting the movie" << std::endl;

// 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "DELETE FROM `"
        << MOVIE_TABLE_NAME << "` WHERE "
        << TITLE_KEY << "=?" and " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete the movie: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

std::cout << "Movie successfully deleted." << std::endl;
return true;
}

//! Create a DynamoDB table.
/*!
 \sa createDynamoDBTable()
 \param tableName: The DynamoDB table's name.
 \param clientConfiguration: Aws client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createDynamoDBTable(const Aws::String &tableName,
                                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

```

```

Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
yearAttributeDefinition.SetAttributeName(YEAR_KEY);
yearAttributeDefinition.SetAttributeType(
    Aws::DynamoDB::Model::ScalarAttributeType::N);
request.AddAttributeDefinitions(yearAttributeDefinition);

Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
yearAttributeDefinition.SetAttributeName(TITLE_KEY);
yearAttributeDefinition.SetAttributeType(
    Aws::DynamoDB::Model::ScalarAttributeType::S);
request.AddAttributeDefinitions(yearAttributeDefinition);

Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
    Aws::DynamoDB::Model::KeyType::HASH);
request.AddKeySchema(yearKeySchema);

Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
    Aws::DynamoDB::Model::KeyType::RANGE);
request.AddKeySchema(titleKeySchema);

Aws::DynamoDB::Model::ProvisionedThroughput throughput;
throughput.WithReadCapacityUnits(
    PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
    PROVISIONED_THROUGHPUT_UNITS);
request.SetProvisionedThroughput(throughput);
request.SetTableName(MOVIE_TABLE_NAME);

std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
const Aws::DynamoDB::Model::CreateTableOutcome &result =
dynamoClient.CreateTable(
    request);
if (!result.IsSuccess()) {
    if (result.GetError().GetErrorCode() ==
        Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
        std::cout << "Table already exists." << std::endl;
        movieTableAlreadyExisted = true;
    }
    else {
        std::cerr << "Failed to create table: "
        << result.GetError().GetMessage();
        return false;
    }
}
// Wait for table to become active.
if (!movieTableAlreadyExisted) {
    std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
    << "' to become active...." << std::endl;
    if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
clientConfiguration)) {
        return false;
    }
    std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
    << std::endl;
}

return true;
}

//! Delete a DynamoDB table.
/*!
 \sa deleteDynamoTable()

```

```

    \param tableName: The DynamoDB table's name.
    \param clientConfiguration: Aws client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteDynamoTable(const Aws::String &tableName,
                                            const Aws::Client::ClientConfiguration
                                            &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
        dynamoClient.DeleteTable(
            request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
            << result.GetResult().GetTableDescription().GetTableName()
            << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
            << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
    \sa waitTableActive()
    \param waitTableActive: The DynamoDB table's name.
    \param clientConfiguration: Aws client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                         const Aws::Client::ClientConfiguration
                                         &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
            dynamoClient.DescribeTable(
                request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
                result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
}

```

```
        }
    return false;
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for C++ API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a struct that is a receiver for methods that can run PartiQL statements.

```
// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
// the
// specified table.
type PartiQLRunner struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// AddMovie runs a PartiQL INSERT statement to add a movie to the DynamoDB table.
func (runner PartiQLRunner) AddMovie(movie Movie) error {
    params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
    movie.Info})
    if err != nil {
        panic(err)
    }
    _, err = runner.DynamoDbClient.ExecuteStatement(context.TODO(),
    &dynamodb.ExecuteStatementInput{
        Statement: aws.String(
            fmt.Sprintf("INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
            runner.TableName)),
        Parameters: params,
    })
    if err != nil {
        log.Printf("Couldn't insert an item with PartiQL. Here's why: %v\n", err)
    }
    return err
}

// GetMovie runs a PartiQL SELECT statement to get a movie from the DynamoDB table
// by
// title and year.
func (runner PartiQLRunner) GetMovie(title string, year int) (Movie, error) {
    var movie Movie
    params, err := attributevalue.MarshalList([]interface{}{title, year})
    if err != nil {
        panic(err)
    }
    response, err := runner.DynamoDbClient.ExecuteStatement(context.TODO(),
    &dynamodb.ExecuteStatementInput{
        Statement: aws.String(
            fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
            runner.TableName)),
        Parameters: params,
    })
    if err != nil {
        log.Printf("Couldn't get a movie with PartiQL. Here's why: %v\n", err)
    }
    err = attributevalue.Unmarshal(response.Items, &movie)
    if err != nil {
        log.Printf("Error unmarshaling movie: %v\n", err)
    }
    return movie, nil
}
```

```
        runner.TableName)),
    Parameters: params,
})
if err != nil {
    log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
} else {
    err = attributevalue.UnmarshalMap(response.Items[0], &movie)
    if err != nil {
        log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
    }
}
return movie, err
}

// UpdateMovie runs a PartiQL UPDATE statement to update the rating of a movie that
// already exists in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovie(movie Movie, rating float64) error {
    params, err := attributevalue.MarshalList([]interface{}{rating, movie.Title,
    movie.Year})
    if err != nil {
        panic(err)
    }
    _, err = runner.DynamoDbClient.ExecuteStatement(context.TODO(),
    &dynamodb.ExecuteStatementInput{
        Statement: aws.String(
            fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
            runner.TableName)),
        Parameters: params,
    })
    if err != nil {
        log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
    }
    return err
}

// DeleteMovie runs a PartiQL DELETE statement to remove a movie from the DynamoDB
// table.
func (runner PartiQLRunner) DeleteMovie(movie Movie) error {
    params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
    if err != nil {
        panic(err)
    }
    _, err = runner.DynamoDbClient.ExecuteStatement(context.TODO(),
    &dynamodb.ExecuteStatementInput{
        Statement: aws.String(
            fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
            runner.TableName)),
        Parameters: params,
    })
    if err != nil {
        log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
        err)
    }
    return err
}
```

Run a scenario that creates a table and runs PartiQL queries.

```
// RunPartiQLSingleScenario shows you how to use the AWS SDK for Go
// to use PartiQL to query a table that stores data about movies.
//
// * Use PartiQL statements to add, get, update, and delete data for individual
//   movies.
//
// This example creates an Amazon DynamoDB service client from the specified
// sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// This example creates and deletes a DynamoDB table to use during the scenario.
func RunPartiQLSingleScenario(sdkConfig aws.Config, tableName string) {
    defer func() {
        if r := recover(); r != nil {
            fmt.Printf("Something went wrong with the demo.")
        }
    }()
}

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon DynamoDB PartiQL single action demo.")
log.Println(strings.Repeat("-", 88))

tableBasics := actions.TableBasics{
    DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
    TableName:      tableName,
}
runner := actions.PartiQLRunner{
    DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
    TableName:      tableName,
}

exists, err := tableBasics.TableExists()
if err != nil {
    panic(err)
}
if !exists {
    log.Printf("Creating table %v...\n", tableName)
    _, err = tableBasics.CreateMovieTable()
    if err != nil {
        panic(err)
    } else {
        log.Printf("Created table %v.\n", tableName)
    }
} else {
    log.Printf("Table %v already exists.\n", tableName)
}
log.Println(strings.Repeat("-", 88))

currentYear, _, _ := time.Now().Date()
customMovie := actions.Movie{
    Title: "24 Hour PartiQL People",
    Year: currentYear,
    Info: map[string]interface{}{
        "plot": "A group of data developers discover a new query language they can't
stop using.",
        "rating": 9.9,
    },
}

log.Printf("Inserting movie '%v' released in %v.", customMovie.Title,
customMovie.Year)
err = runner.AddMovie(customMovie)
if err == nil {
    log.Printf("Added %v to the movie table.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))
```

```
log.Printf("Getting data for movie '%v' released in %v.", customMovie.Title,
customMovie.Year)
movie, err := runner.GetMovie(customMovie.Title, customMovie.Year)
if err == nil {
    log.Println(movie)
}
log.Println(strings.Repeat("-", 88))

newRating := 6.6
log.Printf("Updating movie '%v' with a rating of %v.", customMovie.Title,
newRating)
err = runner.UpdateMovie(customMovie, newRating)
if err == nil {
    log.Printf("Updated %v with a new rating.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Getting data again to verify the update.")
movie, err = runner.GetMovie(customMovie.Title, customMovie.Year)
if err == nil {
    log.Println(movie)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Deleting movie '%v'.\n", customMovie.Title)
err = runner.DeleteMovie(customMovie)
if err == nil {
    log.Printf("Deleted %v.\n", customMovie.Title)
}

err = tableBasics.DeleteTable()
if err == nil {
    log.Printf("Deleted table %v.\n", tableBasics.TableName)
}

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public class ScenarioPartiQ {

    public static void main(String [] args) throws IOException {

        final String usage = "\n" +
            "Usage:\n" +
            "    <fileName>\n\n" +
            "Where:\n" +
            "    fileName - The path to the moviedata.json file that you can
download from the Amazon DynamoDB Developer Guide.\n" ;
```

```
if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String fileName = args[0];
String tableName = "MoviesPartiQ";
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .credentialsProvider(credentialsProvider)
    .region(region)
    .build();

System.out.println("***** Creating an Amazon DynamoDB table named
MoviesPartiQ with a key named year and a sort key named title.");
createTable(ddb, tableName);

System.out.println("***** Loading data into the MoviesPartiQ table.");
loadData(ddb, fileName);

System.out.println("***** Getting data from the MoviesPartiQ table.");
getItem(ddb);

System.out.println("***** Putting a record into the MoviesPartiQ table.");
putRecord(ddb);

System.out.println("***** Updating a record.");
updateTableItem(ddb);

System.out.println("***** Querying the movies released in 2013.");
queryTable(ddb);

System.out.println("***** Deleting the Amazon DynamoDB table.");
deleteDynamoDBTable(ddb, tableName);
ddb.close();
}

public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
    KeySchemaElement key = KeySchemaElement.builder()
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE) // Sort
        .build();
}
```

```
// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(new Long(10))
        .writeCapacityUnits(new Long(10))
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
    dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " +newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String fileName) throws
IOException {

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?, 'info' : ?}";
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0 ;
    List<AttributeValue> parameters = new ArrayList<>();
    while (iter.hasNext()) {

        // Add 200 movies to the table.
        if (t == 200)
            break ;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf(year))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s(title)
            .build();

        AttributeValue att3 = AttributeValue.builder()
```

```
        .s(info)
        .build();

    parameters.add(att1);
    parameters.add(att2);
    parameters.add(att3);

    // Insert the movie into the Amazon DynamoDB table.
    executeStatementRequest(ddb, sqlStatement, parameters);
    System.out.println("Added Movie " +title);

    parameters.remove(att1);
    parameters.remove(att2);
    parameters.remove(att3);
    t++;
}
}

public static void getItem(DynamoDbClient ddb) {

    String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and
title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n("2012")
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("The Perks of Being a Wallflower")
        .build();

    parameters.add(att1);
    parameters.add(att2);

    try {
        ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
        System.out.println("ExecuteStatement successful: "+
response.toString());

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void putRecord(DynamoDbClient ddb) {

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
    try {
        List<AttributeValue> parameters = new ArrayList<>();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2020"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie")
            .build();

        AttributeValue att3 = AttributeValue.builder()
            .s("No Information")
            .build();

        parameters.add(att1);
```

```
parameters.add(att2);
parameters.add(att3);

executeStatementRequest(ddb, sqlStatement, parameters);
System.out.println("Added new movie.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void updateTableItem(DynamoDbClient ddb){

    String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\':";
    ["Merian C. Cooper", "Ernest B. Schoedsack' where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2013"))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("The East")
        .build();

    parameters.add(att1);
    parameters.add(att2);

    try {
        executeStatementRequest(ddb, sqlStatement, parameters);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Item was updated!");
}

// Query the table where the year is 2013.
public static void queryTable(DynamoDbClient ddb) {
    String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
year";
    try {

        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2013"))
            .build();
        parameters.add(att1);

        // Get items in the table and write out the ID value.
        ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
        System.out.println("ExecuteStatement successful: "+
response.toString());

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
```

```
.build();

try {
    ddb.deleteTable(request);

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println(tableName +" was successfully deleted!");
}

private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
ddb, String statement, List<AttributeValue> parameters ) {
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .statement(statement)
        .parameters(parameters)
        .build();

    return ddb.executeStatement(request);
}

private static void processResults(ExecuteStatementResponse
executeStatementResult) {
    System.out.println("ExecuteStatement successful: "+
executeStatementResult.toString());
}
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
export const REGION = "eu-west-1"; // For example, "us-east-1".
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: REGION });
```

Create the document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: false, // false, by default.
    // Whether to convert typeof object to map attribute.
```

```
    convertClassInstanceToMap: false, // false, by default.  
};  
  
const unmarshallOptions = {  
    // Whether to return numbers as a string instead of converting them to native  
    // JavaScript numbers.  
    wrapNumbers: false, // false, by default.  
};  
  
const translateConfig = { marshallOptions, unmarshallOptions };  
  
// Create the DynamoDB document client.  
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, translateConfig);  
  
export { ddbDocClient };
```

Query single items.

```
/*  
import fs from "fs";  
// A practical functional library used to split the data into segments.  
import * as R from "ramda";  
import { ddbClient } from "../libs/ddbClient.js";  
import { ddbDocClient } from "../libs/ddbDocClient.js";  
import { BatchWriteCommand } from "@aws-sdk/lib-dynamodb";  
import {  
    CreateTableCommand,  
    ExecuteStatementCommand,  
} from "@aws-sdk/client-dynamodb";  
if (process.argv.length < 6) {  
    console.log(  
        "Usage: node partiQL_basics.js <tableName> <movieYear1> <movieTitle1>  
<producer1>\n" +  
        "Example: node partiQL_basics.js Movies 2006 'The Departed' 'New View Films'"  
    );  
}  
  
// Helper function to delay running the code while the AWS service calls wait for  
// responses.  
function wait(ms) {  
    var start = Date.now();  
    var end = start;  
    while (end < start + ms) {  
        end = Date.now();  
    }  
}  
  
const tableName = process.argv[2];  
const movieTitle1 = process.argv[3];  
const movieYear1 = process.argv[4];  
const producer1 = process.argv[5];  
  
export const run = async (tableName, movieYear1, movieTitle1, producer1) => {  
    try {  
        console.log("Creating table ...");  
        // Set the parameters.  
        const params = {  
            AttributeDefinitions: [  
                {  
                    AttributeName: "title",  
                    AttributeType: "S",  
                },  
                {  
                    AttributeName: "year",  
                },  
            ],  
            KeySchema: [{  
                AttributeName: "year",  
                KeyType: "HASH",  
            },  
            {  
                AttributeName: "title",  
                KeyType: "RANGE",  
            }],  
            ProvisionedThroughput: {  
                ReadCapacityUnits: 5,  
                WriteCapacityUnits: 5,  
            },  
        };  
        const response = await ddbDocClient.createTable(params).promise();  
        console.log(`Table ${tableName} created successfully!`);  
    } catch (error) {  
        console.error(`Error creating table ${tableName}: ${error.message}`);  
    }  
};
```

```

        AttributeType: "N",
    },
],
KeySchema: [
{
    AttributeName: "title",
    KeyType: "HASH",
},
{
    AttributeName: "year",
    KeyType: "RANGE",
},
],
ProvisionedThroughput: {
    ReadCapacityUnits: 5,
    WriteCapacityUnits: 5,
},
TableName: tableName,
};

const data = await ddbClient.send(new CreateTableCommand(params));
console.log("Waiting for table to be created...");
wait(10000);
console.log(
    "Table created. Table name is ",
    data.TableDescription.TableName
);
try {
    // Before you run this example, download 'movies.json' from https://
docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Js.02.html,
    // and put it in the same folder as the example.
    // Get the movie data parse to convert into a JSON object.
    const allMovies = JSON.parse(fs.readFileSync("moviedata.json", "utf8"));
    // Split the table into segments of 25.
    const dataSegments = R.splitEvery(25, allMovies);
    // Loop batch write operation 10 times to upload 250 items.
    console.log("Writing movies in batch to table...");
    for (let i = 0; i < 10; i++) {
        const segment = dataSegments[i];
        for (let j = 0; j < 25; j++) {
            const params = {
                RequestItems: [
                    [tableName]: [
                        {
                            // Destination Amazon DynamoDB table name.
                            PutRequest: {
                                Item: {
                                    year: segment[j].year,
                                    title: segment[j].title,
                                    info: segment[j].info,
                                },
                            },
                        ],
                    ],
                ];
            };
            ddbDocClient.send(new BatchWriteCommand(params));
        }
    }
    wait(20000);
    console.log("Success, movies written to table.");
    try {
        const params = {
            Statement: "SELECT * FROM " + tableName + " where title=?",
            Parameters: [{ S: movieTitle1 }],
        };
        console.log("Getting movie....");
    }
}

```

```
        console.log("Statement", params.Statement);
        const data = await ddbDocClient.send(
            new ExecuteStatementCommand(params)
        );
        for (let i = 0; i < data.Items.length; i++) {
            console.log(
                "Success. The query return the following data. Item " + i,
                data.Items[i].year,
                data.Items[i].title,
                data.Items[i].info
            );
        }
    }
    try {
        const params = {
            Statement: "DELETE FROM " + tableName + " where title=? and year=?",
            Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
        };
        await ddbDocClient.send(
            new ExecuteStatementCommand(params)
        );
        console.log("Success. Item deleted.");
    }
    try {
        const params = {
            Statement:
                "INSERT INTO " + tableName + " value {'title':?, 'year':?}",
            Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
        };
        await ddbDocClient.send(
            new ExecuteStatementCommand(params)
        );
        console.log("Success. Item added.");
    }
    try {
        const params = {
            Statement:
                "UPDATE " +
                tableName +
                " SET Producer=? where title=? and year=?",
            Parameters: [
                { S: producer1 },
                { S: movieTitle1 },
                { N: movieYear1 },
            ],
        };
        console.log("Updating a single movie...");
        await ddbDocClient.send(
            new ExecuteStatementCommand(params)
        );
        console.log("Success. Item updated.");
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.log("Error updating item. ", err);
    }
    } catch (err) {
        console.log("Error adding items to table. ", err);
    }
} catch (err) {
    console.log("Error deleting movie. ", err);
}
} catch (err) {
    console.log("Error getting movie. ", err);
}
} catch (err) {
    console.log("Error adding movies by batch. ", err);
}
```

```
        } catch (err) {
            console.log("Error creating table. ", err);
        }
    };
    run(tableName, movieYear1, movieTitle1, producer1);
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun main(args: Array<String>) {

    val usage = """
        Usage:
        <fileName>

        Where:
        fileName - The path to the moviedata.json you can download from the
        Amazon DynamoDB Developer Guide.
        """

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    val ddb = DynamoDbClient { region = "us-east-1" }
    val tableName = "MoviesPartiQ"

    // Get the moviedata.json from the Amazon DynamoDB Developer Guide.
    val fileName = args[0]
    println("Creating an Amazon DynamoDB table named MoviesPartiQ with a key named
    id and a sort key named title.")
    createTablePartiQL(ddb, tableName, "year")
    loadDataPartiQL(ddb, fileName)

    println("***** Getting data from the MoviesPartiQ table.")
    getMoviePartiQL(ddb)

    println("***** Putting a record into the MoviesPartiQ table.")
    putRecordPartiQL(ddb)

    println("***** Updating a record.")
    updateTableItemPartiQL(ddb)

    println("***** Querying the movies released in 2013.")
    queryTablePartiQL(ddb)

    println("***** Deleting the MoviesPartiQ table.")
    deleteTablePartiQL(tableName)
}
```

```
suspend fun createTablePartiQL(ddb: DynamoDbClient, tableNameVal: String, key: String) {

    val attDef = AttributeDefinition {
        attributeName = key
        attributeType = ScalarAttributeType.N
    }

    val attDef1 = AttributeDefinition {
        attributeName = "title"
        attributeType = ScalarAttributeType.S
    }

    val keySchemaVal = KeySchemaElement {
        attributeName = key
        keyType = KeyType.Hash
    }

    val keySchemaVal1 = KeySchemaElement {
        attributeName = "title"
        keyType = KeyType.Range
    }

    val provisionedVal = ProvisionedThroughput {
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }

    val request = CreateTableRequest {
        attributeDefinitions = listOf(attDef, attDef1)
        keySchema = listOf(keySchemaVal, keySchemaVal1)
        provisionedThroughput = provisionedVal
        tableName = tableNameVal
    }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists { // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
    ${response.tableDescription?.tableArn}")
}

suspend fun loadDataPartiQL(ddb: DynamoDbClient, fileName: String) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?, 'info' : ?}"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
    var t = 0

    while (iter.hasNext()) {

        if (t == 200)
            break

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
        parameters.add(AttributeValue.N(year.toString()))
        parameters.add(AttributeValue.S(title))
        parameters.add(AttributeValue.S(info))
    }
}
```

```

        executeStatementPartiQL(ddb, sqlStatement, parameters)
        println("Added Movie $title")
        parameters.clear()
        t++
    }
}

suspend fun getMoviePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?"
    val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
    parameters.add(AttributeValue.N("2012"))
    parameters.add(AttributeValue.S("The Perks of Being a Wallflower"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun putRecordPartiQL(ddb: DynamoDbClient) {

    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?, 'info' : ?}"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2020"))
    parameters.add(AttributeValue.S("My Movie"))
    parameters.add(AttributeValue.S("No Info"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Added new movie.")
}

suspend fun updateTableItemPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C. Cooper\", \"Ernest B. Schoedsack\"]' where year=? and title=?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    parameters.add(AttributeValue.S("The East"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Item was updated!")
}

// Query the table where the year is 2013.
suspend fun queryTablePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year = ?"

    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun deleteTablePartiQL(tableNameVal: String) {

    val request = DeleteTableRequest {
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun executeStatementPartiQL(
    ddb: DynamoDbClient,
    statementVal: String,
    parametersVal: List<AttributeValue>
): ExecuteStatementResponse {

```

```
    val request = ExecuteStatementRequest {
        statement = statementVal
        parameters = parametersVal
    }

    return ddb.executeStatement(request)
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

class GettingStartedWithPartiQL
{
    public function run()
    {
        echo("-----\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using
PHP!\n");
        echo("-----\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_$uuid";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
$tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [

```

```

        'year' => [
            'N' => "$movieYear",
        ],
        'title' => [
            'S' => $movieName,
        ],
    ],
];
list($statement, $parameters) = $service->buildStatementAndParameters("INSERT", $tableName, $key);
$service->insertItemByPartiQL($statement, $parameters);

echo "How would you rate the movie from 1-10?\n";
$rating = 0;
while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 || $rating > 10) {
    $rating = testable_readline("Rating (1-10): ");
}
echo "What was the movie about?\n";
while (empty($plot)) {
    $plot = testable_readline("Plot summary: ");
}
$attributes = [
    new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
    new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
];
list($statement, $parameters) = $service->buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
$service->updateItemByPartiQL($statement, $parameters);
echo "Movie added and updated.\n";

$batch = json_decode(loadMovieData());
$service->writeBatch($tableName, $batch);

$movie = $service->getItemByPartiQL($tableName, $key);
echo "\nThe movie {$movie['Items'][0]['title']['S']} was released in
{$movie['Items'][0]['year']['N']}.\n";
echo "What rating would you like to give {$movie['Items'][0]['title']}['S']?\n";
$rating = 0;
while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 || $rating > 10) {
    $rating = testable_readline("Rating (1-10): ");
}
$attributes = [
    new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
    new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
];
list($statement, $parameters) = $service->buildStatementAndParameters("UPDATE", $tableName, $key, $attributes);
$service->updateItemByPartiQL($statement, $parameters);

$movie = $service->getItemByPartiQL($tableName, $key);
echo "Okay, you have rated {$movie['Items'][0]['title']['S']} as a
{$movie['Items'][0]['rating']['N']}.\n";

$service->deleteItemByPartiQL($statement, $parameters);
echo "But, bad news, this was a trap. That movie has now been deleted
because of your rating...harsh.\n";

echo "That's okay though. The book was better. Now, for something lighter,
in what year were you born?\n";

```

```

$birthYear = "not a number";
while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
    $birthYear = testable_readline("Birth year: ");
}
$birthKey = [
    'Key' => [
        'year' => [
            'N' => "$birthYear",
        ],
    ],
];
$result = $service->query($tableName, $birthKey);
$marshal = new Marshaler();
echo "Here are the movies in our collection released the year you were
born:\n";
$oops = "Oops! There were no movies released in that year (that we know
of).\n";
$display = "";
foreach ($result['Items'] as $movie) {
    $movie = $marshal->unmarshalItem($movie);
    $display .= $movie['title'] . "\n";
}
echo ($display) ?: $oops;

$yearsKey = [
    'Key' => [
        'year' => [
            'N' => [
                'minRange' => 1990,
                'maxRange' => 1999,
            ],
        ],
    ],
];
$filter = "year between 1990 and 1999";
echo "\nHere's a list of all the movies released in the 90s:\n";
$result = $service->scan($tableName, $yearsKey, $filter);
foreach ($result['Items'] as $movie) {
    $movie = $marshal->unmarshalItem($movie);
    echo $movie['title'] . "\n";
}

echo "\nCleaning up this demo by deleting table $tableName...\n";
 getService->deleteTable($tableName);
}

public function insertItemByPartiQL(string $statement, array $parameters)
{
    $this->dynamoDbClient->executeStatement([
        'Statement' => "$statement",
        'Parameters' => $parameters,
    ]);
}

public function getItemByPartiQL(string $tableName, array $key): Result
{
    list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);

    return $this->dynamoDbClient->executeStatement([
        'Parameters' => $parameters,
        'Statement' => $statement,
    ]);
}

```

```
public function updateItemByPartiQL(string $statement, array $parameters)
{
    $this->dynamoDbClient->executeStatement([
        'Statement' => $statement,
        'Parameters' => $parameters,
    ]);
}

public function deleteItemByPartiQL(string $statement, array $parameters)
{
    $this->dynamoDbClient->executeStatement([
        'Statement' => $statement,
        'Parameters' => $parameters,
    ]);
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that can run PartiQL statements.

```
from datetime import datetime
from decimal import Decimal
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

from scaffold import Scaffold

logger = logging.getLogger(__name__)

class PartiQLWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource

    def run_partiql(self, statement, params):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because the
        resource transforms input and output from plain old Python objects (POPOs)
        to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statement: The PartiQL statement.
        :param params: The list of PartiQL parameters. These are applied to the
                      statement in the order they are listed.
        :return: The items returned from the statement, if any.
        """

```

```
"""
try:
    output = self.dyn_resource.meta.client.execute_statement(
        Statement=statement, Parameters=params)
except ClientError as err:
    if err.response['Error']['Code'] == 'ResourceNotFoundException':
        logger.error(
            "Couldn't execute PartiQL '%s' because the table does not
exist.", statement)
    else:
        logger.error(
            "Couldn't execute PartiQL '%s'. Here's why: %s: %s", statement,
            err.response['Error']['Code'], err.response['Error']
        ['Message']))
        raise
else:
    return output
```

Run a scenario that creates a table and runs PartiQL queries.

```
def run_scenario(scaffold, wrapper, table_name):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the Amazon DynamoDB PartiQL single statement demo.")
    print('*'*88)

    print(f"Creating table '{table_name}' for the demo...")
    scaffold.create_table(table_name)
    print('*'*88)

    title = "24 Hour PartiQL People"
    year = datetime.now().year
    plot = "A group of data developers discover a new query language they can't
stop using."
    rating = Decimal('9.9')

    print(f"Inserting movie '{title}' released in {year}.")
    wrapper.run_partiql(
        f"INSERT INTO \"{table_name}\" VALUE {{'title': ?, 'year': ?, 'info': ?}}",
        [title, year, {'plot': plot, 'rating': rating}])
    print("Success!")
    print('*'*88)

    print(f"Getting data for movie '{title}' released in {year}.")
    output = wrapper.run_partiql(
        f"SELECT * FROM \"{table_name}\" WHERE title=? AND year=?", [title, year])
    for item in output['Items']:
        print(f"\n{item['title']}, {item['year']}")
        pprint(output['Items'])
    print('*'*88)

    rating = Decimal('2.4')
    print(f"Updating movie '{title}' with a rating of {float(rating)}.")
    wrapper.run_partiql(
        f"UPDATE \"{table_name}\" SET info.rating=? WHERE title=? AND year=?",
        [rating, title, year])
    print("Success!")
    print('*'*88)

    print(f"Getting data again to verify our update.")
    output = wrapper.run_partiql(
        f"SELECT * FROM \"{table_name}\" WHERE title=? AND year=?", [title, year])
```

```
for item in output['Items']:
    print(f"\n{item['title']}, {item['year']}")  
    pprint(output['Items'])  
print('*'*88)  
  
print(f"Deleting movie '{title}' released in {year}.")  
wrapper.run_partiql(  
    f"DELETE FROM \'{table_name}\' WHERE title=? AND year=?", [title, year])
print("Success!")
print('*'*88)  
  
print(f"Deleting table '{table_name}'...")
scaffold.delete_table()
print('*'*88)  
  
print("\nThanks for watching!")
print('*'*88)  
  
if __name__ == '__main__':
    try:
        dyn_res = boto3.resource('dynamodb')
        scaffold = Scaffold(dyn_res)
        movies = PartiQLWrapper(dyn_res)
        run_scenario(scaffold, movies, 'doc-example-table-partiql-movies')
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Python (Boto3) API Reference*.

## Cross-service examples for DynamoDB using AWS SDKs

The following code examples show how to use Amazon DynamoDB with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Build an application to submit data to a DynamoDB table \(p. 594\)](#)
- [Create an API Gateway REST API to track COVID-19 data \(p. 595\)](#)
- [Create a messenger application with Step Functions \(p. 596\)](#)
- [Create a web application to track DynamoDB data \(p. 597\)](#)
- [Create a websocket chat application with API Gateway \(p. 598\)](#)
- [Detect PPE in images with Amazon Rekognition using an AWS SDK \(p. 599\)](#)
- [Invoke a Lambda function from a browser \(p. 599\)](#)
- [Save EXIF and other image information using an AWS SDK \(p. 600\)](#)
- [Use API Gateway to invoke a Lambda function \(p. 601\)](#)
- [Use Step Functions to invoke Lambda functions \(p. 601\)](#)
- [Use scheduled events to invoke a Lambda function \(p. 602\)](#)

### Build an application to submit data to a DynamoDB table

The following code examples show how to build an application that submits data to an Amazon DynamoDB table and notifies you when a user updates the table.

## Java

### SDK for Java 2.x

Shows how to create a dynamic web application that submits data using the Amazon DynamoDB Java API and sends a text message using the Amazon Simple Notification Service Java API.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Amazon SNS

## JavaScript

### SDK for JavaScript V3

This example shows how to build an app that enables users to submit data to an Amazon DynamoDB table, and send a text message to the administrator using Amazon Simple Notification Service (Amazon SNS).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

#### Services used in this example

- DynamoDB
- Amazon SNS

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to create a native Android application that submits data using the Amazon DynamoDB Kotlin API and sends a text message using the Amazon SNS Kotlin API.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Amazon SNS

## Create an API Gateway REST API to track COVID-19 data

The following code example shows how to create a REST API that simulates a system to track daily cases of COVID-19 in the United States, using fictional data.

## Python

### SDK for Python (Boto3)

Shows how to use AWS Chalice with the AWS SDK for Python (Boto3) to create a serverless REST API that uses Amazon API Gateway, AWS Lambda, and Amazon DynamoDB. The REST API simulates a system that tracks daily cases of COVID-19 in the United States, using fictional data. Learn how to:

- Use AWS Chalice to define routes in Lambda functions that are called to handle REST requests that come through API Gateway.
- Use Lambda functions to retrieve and store data in a DynamoDB table to serve REST requests.
- Define table structure and security role resources in an AWS CloudFormation template.
- Use AWS Chalice and CloudFormation to package and deploy all necessary resources.
- Use CloudFormation to clean up all created resources.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- API Gateway
- AWS CloudFormation
- DynamoDB
- Lambda

## Create a messenger application with Step Functions

The following code example shows how to create an AWS Step Functions messenger application that retrieves message records from a database table.

## Python

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with AWS Step Functions to create a messenger application that retrieves message records from an Amazon DynamoDB table and sends them with Amazon Simple Queue Service (Amazon SQS). The state machine integrates with an AWS Lambda function to scan the database for unsent messages.

- Create a state machine that retrieves and updates message records from an Amazon DynamoDB table.
- Update the state machine definition to also send messages to Amazon Simple Queue Service (Amazon SQS).
- Start and stop state machine runs.
- Connect to Lambda, DynamoDB, and Amazon SQS from a state machine by using service integrations.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Lambda
- Amazon SQS
- Step Functions

## Create a web application to track DynamoDB data

The following code examples show how to create a web application that tracks work items in an Amazon DynamoDB table and uses Amazon Simple Email Service (Amazon SES) to send reports.

.NET

### AWS SDK for .NET

Shows how to use the Amazon DynamoDB .NET API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Amazon SES

Java

### SDK for Java 2.x

Shows how to use the Amazon DynamoDB API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Amazon SES

JavaScript

### SDK for JavaScript V3

Shows how to use the Amazon DynamoDB API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Amazon SES

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to use the Amazon DynamoDB API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Amazon SES

Python

#### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) to create a REST service that tracks work items in Amazon DynamoDB and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses the Flask web framework to handle HTTP routing and integrates with a React webpage to present a fully functional web application.

- Build a Flask REST service that integrates with AWS services.
- Read, write, and update work items that are stored in a DynamoDB table.
- Use Amazon SES to send email reports of work items.

For complete source code and instructions on how to set up and run, see the full example in the [AWS Code Examples Repository](#) on GitHub.

#### Services used in this example

- DynamoDB
- Amazon SES

## Create a websocket chat application with API Gateway

The following code example shows how to create a chat application that is served by a websocket API built on Amazon API Gateway.

Python

#### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with Amazon API Gateway V2 to create a websocket API that integrates with AWS Lambda and Amazon DynamoDB.

- Create a websocket API served by API Gateway.
- Define a Lambda handler that stores connections in DynamoDB and posts messages to other chat participants.
- Connect to the websocket chat application and send messages with the Websockets package.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- API Gateway
- DynamoDB
- Lambda

## Detect PPE in images with Amazon Rekognition using an AWS SDK

The following code examples show how to build an app that uses Amazon Rekognition to detect Personal Protective Equipment (PPE) in images.

Java

### SDK for Java 2.x

Shows how to create an AWS Lambda function that detects images with Personal Protective Equipment.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

### SDK for JavaScript V3

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an application to detect personal protective equipment (PPE) in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app saves the results to an Amazon DynamoDB table, and sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for PPE using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Update a DynamoDB table with results.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

## Invoke a Lambda function from a browser

The following code example shows how to invoke an AWS Lambda function from a browser.

## JavaScript

### SDK for JavaScript V2

You can create a browser-based application that uses an AWS Lambda function to update an Amazon DynamoDB table with user selections.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Lambda

### SDK for JavaScript V3

You can create a browser-based application that uses an AWS Lambda function to update an Amazon DynamoDB table with user selections. This app uses AWS SDK for JavaScript v3.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Lambda

## Save EXIF and other image information using an AWS SDK

The following code example shows how to:

- Get EXIF information from a a JPG, JPEG, or PNG file.
- Upload the image file to an Amazon Simple Storage Service (Amazon S3) bucket.
- Use Amazon Rekognition (Amazon Rekognition) to identify the three top attributes (labels in Amazon Rekognition) in the file.
- Add the EXIF and label information to a Amazon DynamoDB (DynamoDB) table in the Region.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

Get EXIF information from a JPG, JPEG, or PNG file, upload the image file to an Amazon Simple Storage Service bucket, use Amazon Rekognition to identify the three top attributes (*labels* in Amazon Rekognition) in the file, and add the EXIF and label information to a Amazon DynamoDB table in the Region.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Amazon Rekognition

- Amazon S3

## Use API Gateway to invoke a Lambda function

The following code examples show how to create an AWS Lambda function invoked by Amazon API Gateway.

Java

### SDK for Java 2.x

Shows how to create an AWS Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create a Lambda function invoked by Amazon API Gateway that scans an Amazon DynamoDB table for work anniversaries and uses Amazon Simple Notification Service (Amazon SNS) to send a text message to your employees that congratulates them at their one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

### SDK for JavaScript V3

Shows how to create an AWS Lambda function by using the Lambda JavaScript runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create a Lambda function invoked by Amazon API Gateway that scans an Amazon DynamoDB table for work anniversaries and uses Amazon Simple Notification Service (Amazon SNS) to send a text message to your employees that congratulates them at their one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

### Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

## Use Step Functions to invoke Lambda functions

The following code examples show how to create an AWS Step Functions state machine that invokes AWS Lambda functions in sequence.

Java

### SDK for Java 2.x

Shows how to create an AWS serverless workflow by using AWS Step Functions and the AWS SDK for Java 2.x. Each workflow step is implemented using an AWS Lambda function.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

JavaScript

### SDK for JavaScript V3

Shows how to create an AWS serverless workflow by using AWS Step Functions and the AWS SDK for JavaScript. Each workflow step is implemented using an AWS Lambda function.

Lambda is a compute service that enables you to run code without provisioning or managing servers. Step Functions is a serverless orchestration service that lets you combine Lambda functions and other AWS services to build business-critical applications.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

#### Services used in this example

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

## Use scheduled events to invoke a Lambda function

The following code examples show how to create an AWS Lambda function invoked by an Amazon EventBridge scheduled event.

Java

### SDK for Java 2.x

Shows how to create an Amazon EventBridge scheduled event that invokes an AWS Lambda function. Configure EventBridge to use a cron expression to schedule when the Lambda function is invoked. In this example, you create a Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create an app that sends a mobile text message to your employees that congratulates them at the one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

#### SDK for JavaScript V3

Shows how to create an Amazon EventBridge scheduled event that invokes an AWS Lambda function. Configure EventBridge to use a cron expression to schedule when the Lambda function is invoked. In this example, you create a Lambda function by using the Lambda JavaScript runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create an app that sends a mobile text message to your employees that congratulates them at the one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

### Services used in this example

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

## Code examples for Amazon EBS using AWS SDKs

The following code examples show how to use Amazon Elastic Block Store with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon EBS using AWS SDKs \(p. 603\)](#)
  - [Create a Amazon EBS snapshot using an AWS SDK \(p. 604\)](#)
  - [Seal and complete the Amazon EBS snapshot using an AWS SDK \(p. 604\)](#)
  - [Write a block of data to an Amazon EBS snapshot using an AWS SDK \(p. 605\)](#)

## Actions for Amazon EBS using AWS SDKs

The following code examples show how to use Amazon Elastic Block Store with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create a Amazon EBS snapshot using an AWS SDK \(p. 604\)](#)

- [Seal and complete the Amazon EBS snapshot using an AWS SDK \(p. 604\)](#)
- [Write a block of data to an Amazon EBS snapshot using an AWS SDK \(p. 605\)](#)

## Create a Amazon EBS snapshot using an AWS SDK

The following code example shows how to create an Amazon EBS snapshot.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn start(client: &Client, description: &str) -> Result<String, Error> {
    let snapshot = client
        .start_snapshot()
        .description(description)
        .encrypted(false)
        .volume_size(1)
        .send()
        .await?;

    Ok(snapshot.snapshot_id.unwrap())
}
```

- For API details, see [StartSnapshot](#) in *AWS SDK for Rust API reference*.

## Seal and complete the Amazon EBS snapshot using an AWS SDK

The following code example shows how to seal and complete an Amazon EBS snapshot.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn finish(client: &Client, id: &str) -> Result<(), Error> {
    client
        .complete_snapshot()
        .changed_blocks_count(2)
        .snapshot_id(id)
        .send()
        .await?;
```

```
    println!("Snapshot ID {}", id);
    println!("The state is 'completed' when all of the modified blocks have been
transferred to Amazon S3.");
    println!("Use the get-snapshot-state code example to get the state of the
snapshot.");
}

Ok(())
}
```

- For API details, see [CompleteSnapshot](#) in *AWS SDK for Rust API reference*.

## Write a block of data to an Amazon EBS snapshot using an AWS SDK

The following code example shows how to write a block of data to an Amazon EBS snapshot.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn add_block(
    client: &Client,
    id: &str,
    idx: usize,
    block: Vec<u8>,
    checksum: &str,
) -> Result<(), Error> {
    client
        .put_snapshot_block()
        .snapshot_id(id)
        .block_index(idx as i32)
        .block_data(ByteStream::from(block))
        .checksum(checksum)
        .checksum_algorithm(ChecksumAlgorithm::ChecksumAlgorithmSha256)
        .data_length(EBS_BLOCK_SIZE as i32)
        .send()
        .await?;

    Ok(())
}
```

- For API details, see [PutSnapshotBlock](#) in *AWS SDK for Rust API reference*.

## Code examples for Amazon EC2 using AWS SDKs

The following code examples show how to use Amazon Elastic Compute Cloud with an AWS software development kit (SDK).

## Get started

### Hello Amazon EC2

The following code examples show how to get started using Amazon Elastic Compute Cloud (Amazon EC2).

Java

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeEC2SecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
        .groupIds(groupId)
        .build();

        DescribeSecurityGroupsResponse response =
ec2.describeSecurityGroups(request);
        for(SecurityGroup group : response.securityGroups()) {
            System.out.printf(
                "Found Security Group with id %s, " +
                "vpc id %s " +
                "and description %s",
                group.groupId(),
                group.vpcId(),
                group.description());
        }
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for Java 2.x API Reference*.

Python

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import boto3

def hello_ec2(ec2_resource):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Elastic Compute Cloud
    (Amazon EC2) resource and list the security groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

```

```
:param ec2_resource: A Boto3 EC2 ServiceResource object. This object is a high-
level
                           resource that wraps the low-level EC2 service API.
"""
print("Hello, Amazon EC2! Let's list up to 10 of your security groups:")
for sg in ec2_resource.security_groups.limit(10):
    print(f"\t{sg.id}: {sg.group_name}")

if __name__ == '__main__':
    hello_ec2(boto3.resource('ec2'))
```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for Python (Boto3) API Reference*.

## Code examples

- [Actions for Amazon EC2 using AWS SDKs \(p. 608\)](#)
  - [Allocate an Elastic IP address for Amazon EC2 using an AWS SDK \(p. 608\)](#)
  - [Associate an Elastic IP address with an Amazon EC2 instance using an AWS SDK \(p. 610\)](#)
  - [Create an Amazon EC2 VPC using an AWS SDK \(p. 612\)](#)
  - [Create an Amazon EC2 security group using an AWS SDK \(p. 613\)](#)
  - [Create a security key pair for Amazon EC2 using an AWS SDK \(p. 616\)](#)
  - [Create and run an Amazon EC2 instance using an AWS SDK \(p. 619\)](#)
  - [Delete an Amazon EC2 VPC using an AWS SDK \(p. 622\)](#)
  - [Delete an Amazon EC2 security group using an AWS SDK \(p. 623\)](#)
  - [Delete an Amazon EC2 security key pair using an AWS SDK \(p. 625\)](#)
  - [Delete the Amazon EBS snapshot using an AWS SDK \(p. 628\)](#)
  - [Describe the Availability Zones for your account using an AWS SDK \(p. 629\)](#)
  - [Describes the Regions for your account using an AWS SDK \(p. 629\)](#)
  - [Describes the status of Amazon EC2 instances using an AWS SDK \(p. 630\)](#)
  - [Describe Amazon EC2 instances using an AWS SDK \(p. 631\)](#)
  - [Describe one or more Amazon EBS snapshots using an AWS SDK \(p. 636\)](#)
  - [Disassociate an Elastic IP address from an Amazon EC2 instance using an AWS SDK \(p. 637\)](#)
  - [Enables monitoring for a running Amazon EC2 instance using an AWS SDK \(p. 638\)](#)
  - [Get data about Amazon Machine Images using an AWS SDK \(p. 640\)](#)
  - [Get data about an Amazon EC2 security group using an AWS SDK \(p. 641\)](#)
  - [Get data about Amazon EC2 instance types using an AWS SDK \(p. 642\)](#)
  - [Get details about Elastic IP addresses using an AWS SDK \(p. 644\)](#)
  - [List Amazon EC2 security key pairs using an AWS SDK \(p. 645\)](#)
  - [Reboot an Amazon EC2 instance using an AWS SDK \(p. 646\)](#)
  - [Release an Elastic IP address using an AWS SDK \(p. 648\)](#)
  - [Set inbound rules for an Amazon EC2 security group using an AWS SDK \(p. 650\)](#)
  - [Start an Amazon EC2 instance using an AWS SDK \(p. 651\)](#)
  - [Stop an Amazon EC2 instance using an AWS SDK \(p. 655\)](#)
  - [Terminate an Amazon EC2 instance using an AWS SDK \(p. 658\)](#)
- [Scenarios for Amazon EC2 using AWS SDKs \(p. 660\)](#)
  - [Get started with Amazon EC2 instances using an AWS SDK \(p. 660\)](#)
- [Cross-service examples for Amazon EC2 using AWS SDKs \(p. 683\)](#)
  - [Create a long-lived Amazon EMR cluster and run several steps using an AWS SDK \(p. 683\)](#)
  - [Create a short-lived Amazon EMR cluster and run a step using an AWS SDK \(p. 684\)](#)

## Actions for Amazon EC2 using AWS SDKs

The following code examples show how to use Amazon Elastic Compute Cloud with AWS SDKs. Each example calls an individual service function.

### Examples

- [Allocate an Elastic IP address for Amazon EC2 using an AWS SDK \(p. 608\)](#)
- [Associate an Elastic IP address with an Amazon EC2 instance using an AWS SDK \(p. 610\)](#)
- [Create an Amazon EC2 VPC using an AWS SDK \(p. 612\)](#)
- [Create an Amazon EC2 security group using an AWS SDK \(p. 613\)](#)
- [Create a security key pair for Amazon EC2 using an AWS SDK \(p. 616\)](#)
- [Create and run an Amazon EC2 instance using an AWS SDK \(p. 619\)](#)
- [Delete an Amazon EC2 VPC using an AWS SDK \(p. 622\)](#)
- [Delete an Amazon EC2 security group using an AWS SDK \(p. 623\)](#)
- [Delete an Amazon EC2 security key pair using an AWS SDK \(p. 625\)](#)
- [Delete the Amazon EBS snapshot using an AWS SDK \(p. 628\)](#)
- [Describe the Availability Zones for your account using an AWS SDK \(p. 629\)](#)
- [Describes the Regions for your account using an AWS SDK \(p. 629\)](#)
- [Describes the status of Amazon EC2 instances using an AWS SDK \(p. 630\)](#)
- [Describe Amazon EC2 instances using an AWS SDK \(p. 631\)](#)
- [Describe one or more Amazon EBS snapshots using an AWS SDK \(p. 636\)](#)
- [Disassociate an Elastic IP address from an Amazon EC2 instance using an AWS SDK \(p. 637\)](#)
- [Enables monitoring for a running Amazon EC2 instance using an AWS SDK \(p. 638\)](#)
- [Get data about Amazon Machine Images using an AWS SDK \(p. 640\)](#)
- [Get data about an Amazon EC2 security group using an AWS SDK \(p. 641\)](#)
- [Get data about Amazon EC2 instance types using an AWS SDK \(p. 642\)](#)
- [Get details about Elastic IP addresses using an AWS SDK \(p. 644\)](#)
- [List Amazon EC2 security key pairs using an AWS SDK \(p. 645\)](#)
- [Reboot an Amazon EC2 instance using an AWS SDK \(p. 646\)](#)
- [Release an Elastic IP address using an AWS SDK \(p. 648\)](#)
- [Set inbound rules for an Amazon EC2 security group using an AWS SDK \(p. 650\)](#)
- [Start an Amazon EC2 instance using an AWS SDK \(p. 651\)](#)
- [Stop an Amazon EC2 instance using an AWS SDK \(p. 655\)](#)
- [Terminate an Amazon EC2 instance using an AWS SDK \(p. 658\)](#)

## Allocate an Elastic IP address for Amazon EC2 using an AWS SDK

The following code examples show how to allocate an Elastic IP address for Amazon EC2.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String getAllocateAddress( Ec2Client ec2, String instanceId) {  
  
    try {  
        AllocateAddressRequest allocateRequest =  
AllocateAddressRequest.builder()  
            .domain(DomainType.VPC)  
            .build();  
  
        AllocateAddressResponse allocateResponse =  
ec2.allocateAddress(allocateRequest);  
        String allocationId = allocateResponse.allocationId();  
        AssociateAddressRequest associateRequest =  
AssociateAddressRequest.builder()  
            .instanceId(instanceId)  
            .allocationId(allocationId)  
            .build();  
  
        AssociateAddressResponse associateResponse =  
ec2.associateAddress(associateRequest);  
        return associateResponse.associationId();  
  
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [AllocateAddress](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ElasticIpWrapper:  
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address  
actions."""  
    def __init__(self, ec2_resource, elastic_ip=None):  
        """  
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource  
        is used to create additional high-level objects  
        that wrap low-level Amazon EC2 service actions.  
        :param elastic_ip: A Boto3 VpcAddress object. This is a high-level object  
        that  
            wraps Elastic IP actions.  
        """  
        self.ec2_resource = ec2_resource  
        self.elastic_ip = elastic_ip  
  
    @classmethod  
    def from_resource(cls):  
        ec2_resource = boto3.resource('ec2')  
        return cls(ec2_resource)  
  
    def allocate(self):  
        """  
        Allocates an Elastic IP address that can be associated with an Amazon EC2
```

```
        instance. By using an Elastic IP address, you can keep the public IP
address
        constant even when you restart the associated instance.

    :return: The newly created Elastic IP object. By default, the address is
not
            associated with any instance.
    """
    try:
        response = self.ec2_resource.meta.client.allocate_address(Domain='vpc')
        self.elastic_ip =
    self.ec2_resource.VpcAddress(response['AllocationId'])
    except ClientError as err:
        logger.error(
            "Couldn't allocate Elastic IP. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return self.elastic_ip
```

- For API details, see [AllocateAddress](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_ec2->allocateaddress( iv_domain = 'vpc' ).    " oo_result is
    returned for testing purpose "
    MESSAGE 'Allocated an Elastic IP address' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [AllocateAddress](#) in *AWS SDK for SAP ABAP API reference*.

## Associate an Elastic IP address with an Amazon EC2 instance using an AWS SDK

The following code examples show how to associate an Elastic IP address with an Amazon EC2 instance.

### Java

#### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [AssociateAddress](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
actions."""
    def __init__(self, ec2_resource, elastic_ip=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param elastic_ip: A Boto3 VpcAddress object. This is a high-level object
                            that
                            wraps Elastic IP actions.
        """
        self.ec2_resource = ec2_resource
        self.elastic_ip = elastic_ip

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def associate(self, instance):
        """
        Associates an Elastic IP address with an instance. When this association is
        created, the Elastic IP's public IP address is immediately used as the
public
        IP address of the associated instance.

        :param instance: A Boto3 Instance object. This is a high-level object that
wraps
                            Amazon EC2 instance actions.
        """


```

```
:return: A response that contains the ID of the association.  
"""  
if self.elastic_ip is None:  
    logger.info("No Elastic IP to associate.")  
    return  
  
try:  
    response = self.elastic_ip.associate(InstanceId=instance.id)  
except ClientError as err:  
    logger.error(  
        "Couldn't associate Elastic IP %s with instance %s. Here's why: %s:  
        %s",  
        self.elastic_ip.allocation_id, instance.id,  
        err.response['Error']['Code'], err.response['Error']['Message'])  
    raise  
return response
```

- For API details, see [AssociateAddress](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_ec2->associateaddress(                                     " oo_result  
is returned for testing purpose "  
        iv_allocationid = iv_allocation_id  
        iv_instanceid = iv_instance_id  
    ).  
    MESSAGE 'Associated Elastic IP address with an EC2 instance' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.  
        MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- For API details, see [AssociateAddress](#) in *AWS SDK for SAP ABAP API reference*.

## Create an Amazon EC2 VPC using an AWS SDK

The following code example shows how to create an Amazon EC2 VPC.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to create an Amazon Elastic Compute Cloud (Amazon EC2) VPC
/// using the AWS SDK for .NET and .NET Core 5.0.
/// </summary>
public class CreateVPC
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and then calls the
    /// CreateVpcAsync method to create the VPC.
    /// </summary>
    public static async Task Main()
    {
        // If you do not want to create the VPC in the same AWS Region as
        // the default users on your system, you need to supply the AWS
        // Region as a parameter to the client constructor.
        var client = new AmazonEC2Client();

        var response = await client.CreateVpcAsync(new CreateVpcRequest
        {
            CidrBlock = "10.0.0.0/16",
        });

        Vpc vpc = response.Vpc;

        if (vpc is not null)
        {
            Console.WriteLine($"Created VPC with ID: {vpc.VpcId}.");
        }
    }
}
```

- For API details, see [CreateVpc](#) in *AWS SDK for .NET API Reference*.

## Create an Amazon EC2 security group using an AWS SDK

The following code examples show how to create an Amazon EC2 security group.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to create a security group for an Amazon Elastic Compute
/// Cloud (Amazon EC2) VPC using the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
/// </summary>
```

```
/// </summary>
public class CreateSecurityGroup
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and uses the
    /// CreateSecurityGroupAsync method to create the security group.
    /// </summary>
    public static async Task Main()
    {
        string vpcId = "vpc-1234567890abcdefa";
        string vpcDescription = "Sample security group";
        string groupName = "sample-security-group";

        var client = new AmazonEC2Client();
        var response = await client.CreateSecurityGroupAsync(new
CreateSecurityGroupRequest
        {
            Description = vpcDescription,
           GroupName = groupName,
            VpcId = vpcId,
        });

        string groupId = response.GroupId;

        Console.WriteLine($"Successfully created security group: {groupName}
with ID: {groupId}");
    }
}
```

- For API details, see [CreateSecurityGroup](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createEC2SecurityGroup( Ec2Client ec2, String groupName,
String groupDesc, String vpcId) {
    try {

        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
        .groupName(groupName)
        .description(groupDesc)
        .vpcId(vpcId)
        .build();

        CreateSecurityGroupResponse resp=
ec2.createSecurityGroup(createRequest);

        IpRange ipRange = IpRange.builder()
        .cidrIp("0.0.0.0/0").build();

        IpPermission ipPerm = IpPermission.builder()
        .ipProtocol("tcp")
        .toPort(80)
        .fromPort(80)
        .ipRanges(ipRange)
```

```

        .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
            AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(groupName)
                .ipPermissions(ipPerm, ipPerm2)
                .build();

        AuthorizeSecurityGroupIngressResponse authResponse =
            ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.printf("Successfully added ingress policy to Security Group
%s", groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

```

- For API details, see [CreateSecurityGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
    actions."""
    def __init__(self, ec2_resource, security_group=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
            is used to create additional high-level objects
            that wrap low-level Amazon EC2 service actions.
        :param security_group: A Boto3 SecurityGroup object. This is a high-level
            object
            that wraps security group actions.
        """
        self.ec2_resource = ec2_resource
        self.security_group = security_group

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def create(self, group_name, group_description):
        """
        Creates a security group in the default virtual private cloud (VPC) of the
        current account.
        """

```

```
:param group_name: The name of the security group to create.  
:param group_description: The description of the security group to create.  
:return: A Boto3 SecurityGroup object that represents the newly created  
security group.  
"""  
    try:  
        self.security_group = self.ec2_resource.create_security_group(  
            GroupName=group_name, Description=group_description)  
    except ClientError as err:  
        logger.error(  
            "Couldn't create security group %s. Here's why: %s: %s",  
            group_name,  
            err.response['Error']['Code'], err.response['Error']['Message'])  
        raise  
    else:  
        return self.security_group
```

- For API details, see [CreateSecurityGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_ec2->createsecuritygroup(                                     " oo_result is  
    returned for testing purpose"  
        iv_description = 'Security group example'  
        iv_groupname = iv_security_group_name  
        iv_vpcid = iv_vpc_id  
    ).  
    MESSAGE 'Security group created' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
        >av_err_msg }|.  
        MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [CreateSecurityGroup](#) in *AWS SDK for SAP ABAP API reference*.

## Create a security key pair for Amazon EC2 using an AWS SDK

The following code examples show how to create a security key pair for Amazon EC2.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to create a new Amazon Elastic Compute Cloud (Amazon EC2)
/// key pair. The example was uses the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
/// </summary>
public class CreateKeyPair
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and then calls the
    /// CreateKeyPairAsync method to create the new key pair.
    /// </summary>
    public static async Task Main()
    {
        string keyName = "sdk-example-key-pair";

        // If the default user on your system is not the same as
        // the Region where you want to create the key pair, you
        // need to supply the AWS Region as a parameter to the
        // client constructor.
        var client = new AmazonEC2Client();

        var request = new CreateKeyPairRequest
        {
            KeyName = keyName,
        };

        var response = await client.CreateKeyPairAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var kp = response.KeyPair;
            Console.WriteLine($"'{kp.KeyName}' with the ID: {kp.KeyPairId}.");
        }
        else
        {
            Console.WriteLine("Could not create key pair.");
        }
    }
}
```

- For API details, see [CreateKeyPair](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createEC2KeyPair(Ec2Client ec2, String keyName ) {
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
```

```
        .build();

        ec2.createKeyPair(request);
        System.out.printf("Successfully created key pair named %s", keyName);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateKeyPair](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyPairWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions."""
    def __init__(self, ec2_resource, key_file_dir, key_pair=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param key_file_dir: The folder where the private key information is
                            stored.
                            This should be a secure folder.
        :param key_pair: A Boto3 KeyPair object. This is a high-level object that
                        wraps key pair actions.
        """
        self.ec2_resource = ec2_resource
        self.key_pair = key_pair
        self.key_file_path = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource, tempfile.TemporaryDirectory())

    def create(self, key_name):
        """
        Creates a key pair that can be used to securely connect to an EC2 instance.
        The returned key pair contains private key information that cannot be
        retrieved
        again. The private key data is stored as a .pem file.

        :param key_name: The name of the key pair to create.
        :return: A Boto3 KeyPair object that represents the newly created key pair.
        """
        try:
            self.key_pair = self.ec2_resource.create_key_pair(KeyName=key_name)
            self.key_file_path = os.path.join(self.key_file_dir.name,
                                             f'{self.key_pair.name}.pem')
            with open(self.key_file_path, 'w') as key_file:
                key_file.write(self.key_pair.key_material)
        except ClientError as err:
            logger.error(
```

```
"Couldn't create key %s. Here's why: %s: %s", key_name,
    err.response['Error']['Code'], err.response['Error']['Message'])
raise
else:
    return self.key_pair
```

- For API details, see [CreateKeyPair](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_ec2->createkeypair( iv_keyname = iv_key_name ).
        " oo_result is returned for testing purpose "
        MESSAGE 'EC2 key pair created' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateKeyPair](#) in *AWS SDK for SAP ABAP API reference*.

## Create and run an Amazon EC2 instance using an AWS SDK

The following code examples show how to create and run an Amazon EC2 instance.

### Java

#### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createEC2Instance(Ec2Client ec2, String name, String
amiId) {

    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
        .minCount(1)
        .build();

    RunInstancesResponse response = ec2.runInstances(runRequest);
    String instanceId = response.instances().get(0).instanceId();
    Tag tag = Tag.builder()
        .key("Name")
```

```
.value(name)
.build();

CreateTagsRequest tagRequest = CreateTagsRequest.builder()
    .resources(instanceId)
    .tags(tag)
    .build();

try {
    ec2.createTags(tagRequest);
    System.out.printf("Successfully started EC2 Instance %s based on AMI %s", instanceId, amiId);
    return instanceId;
} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
```

- For API details, see [RunInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""
    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object that
                        wraps instance actions.
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def create(
        self, image, instance_type, key_pair, security_groups=None):
        """
        Creates a new EC2 instance. The instance starts immediately after
        it is created.

        The instance is created in the default VPC of the current account.

        :param image: A Boto3 Image object that represents an Amazon Machine Image
                      (AMI)
                                         that defines attributes of the instance that is created. The
                                         AMI
        """
        # Implementation of create method

```

```

        defines things like the kind of operating system and the type
of
        storage used by the instance.
:param instance_type: The type of instance to create, such as 't2.micro'.
        The instance type defines things like the number of
CPUs and
        the amount of memory.
:param key_pair: A Boto3 KeyPair or KeyPairInfo object that represents the
key
        pair that is used to secure connections to the instance.
:param security_groups: A list of Boto3 SecurityGroup objects that
represents the
        security groups that are used to grant access to
the
        instance. When no security groups are specified,
the
        default security group of the VPC is used.
:return: A Boto3 Instance object that represents the newly created
instance.
"""
try:
    instance_params = {
        'ImageId': image.id, 'InstanceType': instance_type, 'KeyName':
key_pair.name
    }
    if security_groups is not None:
        instance_params['SecurityGroupIds'] = [sg.id for sg in
security_groups]
    self.instance = self.ec2_resource.create_instances(**instance_params,
MinCount=1, MaxCount=1)[0]
    self.instance.wait_until_running()
except ClientError as err:
    logging.error(
        "Couldn't create instance with image %s, instance type %s, and key
%s."
        "Here's why: %s: %s", image.id, instance_type, key_pair.name,
err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return self.instance

```

- For API details, see [RunInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

" Create tags for resource created during instance launch "
DATA lt_tagspecifications TYPE /aws1/
cl_ec2tagspecification=>tt_tagspecificationlist.
DATA ls_tagspecifications LIKE LINE OF lt_tagspecifications.
ls_tagspecifications = NEW /aws1/cl_ec2tagspecification(
iv_resourcetype = 'instance'

```

```
    it_tags = VALUE /aws1/cl_ec2tag=>tt_taglist(
        ( NEW /aws1/cl_ec2tag( iv_key = 'Name' iv_value = iv_tag_value ) )
    ).
APPEND ls_tagspecifications TO lt_tagspecifications.

TRY.
    " Create/Launch EC2 instance "
    oo_result = lo_ec2->runinstances(
        returned for testing purpose "
        iv_imageid = iv_ami_id
        iv_instancetype = 't2.micro'
        iv_maxcount = 1
        iv_mincount = 1
        it_tagspecifications = lt_tagspecifications
        iv_subnetid = iv_subnet_id
    ).
    MESSAGE 'EC2 instance created' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [RunInstances](#) in [AWS SDK for SAP ABAP API reference](#).

## Delete an Amazon EC2 VPC using an AWS SDK

The following code example shows how to delete an Amazon EC2 VPC.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to delete an existing Amazon Elastic Compute Cloud
/// (Amazon EC2) VPC. The example was created using the AWS SDK for .NET
/// version 3.7 and .NET Core 5.0.
/// </summary>
public class DeleteVPC
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and then calls the
    /// DeleteVpcAsync method to delete the VPC.
    /// </summary>
    public static async Task Main()
    {
        string vpcId = "vpc-0123456789abc";

        // If your Amazon EC2 VPC is not defined in the same AWS Region as
        // the default AWS user on your system, you need to supply the AWS
        // Region as a parameter to the client constructor.
    }
}
```

```
var client = new AmazonEC2Client();

var request = new DeleteVpcRequest
{
    VpcId = vpcId,
};

var response = await client.DeleteVpcAsync(request);

if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
{
    Console.WriteLine($"Successfully deleted VPC with ID: {vpcId}.");
}
}
```

- For API details, see [DeleteVpc](#) in *AWS SDK for .NET API Reference*.

## Delete an Amazon EC2 security group using an AWS SDK

The following code examples show how to delete an Amazon EC2 security group.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to use Amazon Elastic Compute Cloud (Amazon EC2) and the
/// AWS SDK for .NET to delete an existing security group. This example
/// was created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DeleteSecurityGroup
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and then uses it to call
    /// the DeleteSecurityGroupAsync method to delete the security group.
    /// </summary>
    public static async Task Main()
    {
        string secGroupId = "sg-1234567890abcd";
        string groupName = "sample-security-group";

        // If your Amazon EC2 security group is not defined in the same AWS
        // Region as the default user on your system, you need to supply
        // the AWS Region as a parameter to the client constructor.
        var client = new AmazonEC2Client();

        var request = new DeleteSecurityGroupRequest
        {
            GroupId = secGroupId,
           GroupName = groupName,
        };
    }
}
```

```
        var response = await client.DeleteSecurityGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully deleted {groupName}.");
        }
        else
        {
            Console.WriteLine($"Could not delete {groupName}.");
        }
    }
}
```

- For API details, see [DeleteSecurityGroup](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {

    try {
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.printf("Successfully deleted Security Group with id %s",
groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteSecurityGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
actions."""
    def __init__(self, ec2_resource, security_group=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
```

```
        is used to create additional high-level objects
        that wrap low-level Amazon EC2 service actions.
:param security_group: A Boto3 SecurityGroup object. This is a high-level
object
                           that wraps security group actions.
"""
self.ec2_resource = ec2_resource
self.security_group = security_group

@classmethod
def from_resource(cls):
    ec2_resource = boto3.resource('ec2')
    return cls(ec2_resource)

def delete(self):
    """
    Deletes the security group.
    """
    if self.security_group is None:
        logger.info("No security group to delete.")
        return

    group_id = self.security_group.id
    try:
        self.security_group.delete()
    except ClientError as err:
        logger.error(
            "Couldn't delete security group %s. Here's why: %s: %s",
            group_id,
            err.response['Error']['Code'],
            err.response['Error']['Message']
        )
        raise
```

- For API details, see [DeleteSecurityGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    lo_ec2->deletesecuritygroup( iv_groupid = iv_security_group_id ).
    MESSAGE 'Security group deleted' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DeleteSecurityGroup](#) in *AWS SDK for SAP ABAP API reference*.

## Delete an Amazon EC2 security key pair using an AWS SDK

The following code examples show how to delete an Amazon EC2 security key pair.

.NET

**AWS SDK for .NET**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to delete an existing Amazon Elastic Compute Cloud
/// (Amazon EC2) key pair. The example uses the AWS SDK for .NET
/// version 3.7 and .NET Core 5.0.
/// </summary>
public class DeleteKeyPair
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and then calls the
    /// DeleteKeyPairAsync method to delete the key pair.
    /// </summary>
    public static async Task Main()
    {
        string keyName = "sdk-example-key-pair";

        // If the key pair was not created in the same AWS Region as
        // the default user on your system, you need to supply
        // the AWS Region as a parameter to the client constructor.
        var client = new AmazonEC2Client();

        var request = new DeleteKeyPairRequest
        {
            KeyName = keyName,
        };

        var response = await client.DeleteKeyPairAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully deleted the key pair:
{keyName}.");
        }
        else
        {
            Console.WriteLine("Could not delete the key pair.");
        }
    }
}
```

- For API details, see [DeleteKeyPair](#) in *AWS SDK for .NET API Reference*.

Java

**SDK for Java 2.x**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteKeys(Ec2Client ec2, String keyPair) {  
  
    try {  
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()  
            .keyName(keyPair)  
            .build();  
  
        ec2.deleteKeyPair(request);  
        System.out.printf("Successfully deleted key pair named %s", keyPair);  
  
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteKeyPair](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyPairWrapper:  
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions."""  
    def __init__(self, ec2_resource, key_file_dir, key_pair=None):  
        """  
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource  
            is used to create additional high-level objects  
            that wrap low-level Amazon EC2 service actions.  
        :param key_file_dir: The folder where the private key information is  
            stored.  
            This should be a secure folder.  
        :param key_pair: A Boto3 KeyPair object. This is a high-level object that  
            wraps key pair actions.  
        """  
        self.ec2_resource = ec2_resource  
        self.key_pair = key_pair  
        self.key_file_path = None  
        self.key_file_dir = key_file_dir  
  
    @classmethod  
    def from_resource(cls):  
        ec2_resource = boto3.resource('ec2')  
        return cls(ec2_resource, tempfile.TemporaryDirectory())  
  
    def delete(self):  
        """  
        Deletes a key pair.  
        """  
        if self.key_pair is None:  
            logger.info("No key pair to delete.")  
            return  
  
        key_name = self.key_pair.name  
        try:  
            self.key_pair.delete()
```

```
        self.key_pair = None
    except ClientError as err:
        logger.error(
            "Couldn't delete key %s. Here's why: %s : %s",
            key_name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
```

- For API details, see [DeleteKeyPair](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    lo_ec2->deletekeypair( iv_keyname = iv_key_name ).
    MESSAGE 'EC2 key pair deleted' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DeleteKeyPair](#) in *AWS SDK for SAP ABAP API reference*.

## Delete the Amazon EBS snapshot using an AWS SDK

The following code example shows how to delete an Amazon EBS snapshot.

### Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn delete_snapshot(client: &Client, id: &str) -> Result<(), Error> {
    client.delete_snapshot().snapshot_id(id).send().await?;

    println!("Deleted");

    Ok(())
}
```

- For API details, see [DeleteSnapshot](#) in *AWS SDK for Rust API reference*.

## Describe the Availability Zones for your account using an AWS SDK

The following code example shows how to describe Amazon EC2 Availability Zones.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_ec2->describeavailabilityzones( ) .  
    "oo_result is returned for testing purpose "  
    DATA(lt_zones) = oo_result->get_availabilityzones( ).  
    MESSAGE 'Retrieved information about availability zone(s)' TYPE 'I'.  
  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.  
        MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [DescribeAvailabilityZones](#) in *AWS SDK for SAP ABAP API reference*.

## Describes the Regions for your account using an AWS SDK

The following code examples show how to describe Amazon EC2 Regions.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_regions(client: &Client) -> Result<(), Error> {  
    let rsp = client.describe_regions().send().await?;  
  
    println!("Regions:");  
    for region in rsp.regions().unwrap_or_default() {  
        println!("  {}", region.region_name().unwrap());  
    }  
}
```

```
        }
    Ok(())
}
```

- For API details, see [DescribeRegions](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_ec2->describeregions( ) .                               " oo_result
is returned for testing purpose "
    DATA(lt_regions) = oo_result->get_regions( ).
    MESSAGE 'Retrieved information about region(s)' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DescribeRegions](#) in *AWS SDK for SAP ABAP API reference*.

## Describes the status of Amazon EC2 instances using an AWS SDK

The following code example shows how to describe the status of Amazon EC2 instances.

### Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_all_events(client: &Client) -> Result<(), Error> {
    let resp = client.describe_regions().send().await.unwrap();

    for region in resp.regions.unwrap_or_default() {
        let reg: &'static str =
            Box::leak(Box::from(region.region_name().unwrap()));
    }
}
```

```
let region_provider = RegionProviderChain::default_provider().or_else(reg);
let config = aws_config::from_env().region(region_provider).load().await;
let new_client = Client::new(&config);

let resp = new_client.describe_instance_status().send().await;

println!("Instances in region {}:", reg);
println!();

for status in resp.unwrap().instance_statuses().unwrap_or_default() {
    println!(
        "    Events scheduled for instance ID: {}",
        status.instance_id().unwrap_or_default()
    );
    for event in status.events().unwrap_or_default() {
        println!("        Event ID:      {}",
            event.instance_event_id().unwrap());
        println!("        Description:  {}", event.description().unwrap());
        println!("        Event code:   {}", event.code().unwrap().as_ref());
        println!();
    }
}
Ok(())
}
```

- For API details, see [DescribeInstanceStatus](#) in *AWS SDK for Rust API reference*.

## Describe Amazon EC2 instances using an AWS SDK

The following code examples show how to describe Amazon EC2 instances.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// This example shows how to list your Amazon Elastic Compute Cloud
/// (Amazon EC2) instances. It was created using the AWS SDK for .NET
/// version 3.7 and .NET Core 5.0.
/// </summary>
public class DescribeInstances
{
    /// <summary>
    /// The Main method creates the Amazon EC2 client object and then calls
    /// first GetInstanceDescriptions and then GetInstanceDescriptionsFiltered
    /// to display the list of Amazon EC2 Instances attached to the default
    /// account.
    /// </summary>
    public static async Task Main()
```

```
{
    // If the Region of the EC2 instances you want to list is different
    // from the default user's Region, you need to specify the Region
    // when you create the client object.
    // For example: RegionEndpoint.USWest1.
    var eC2Client = new AmazonEC2Client();

    // List all EC2 instances.
    await GetInstanceDescriptions(eC2Client);

    string tagName = "IncludeInList";
    string tagValue = "Yes";
    await GetInstanceDescriptionsFiltered(eC2Client, tagName, tagValue);
}

/// <summary>
/// This method uses a paginator to list all of the EC2 Instances
/// attached to the default account.
/// </summary>
/// <param name="client">The Amazon EC2 client object used to call
/// the DescribeInstances method.</param>
public static async Task GetInstanceDescriptions(AmazonEC2Client client)
{
    var request = new DescribeInstancesRequest();

    Console.WriteLine("Showing all instances:");
    var paginator = client.Paginator.DescribeInstances(request);

    await foreach (var response in paginator.Responses)
    {
        foreach (var reservation in response.Reservations)
        {
            foreach (var instance in reservation.Instances)
            {
                Console.Write($"Instance ID: {instance.InstanceId}");
                Console.WriteLine($"\\tCurrent State:
{instance.State.Name}");
            }
        }
    }
}

/// <summary>
/// This method lists the EC2 instances for this account which have set
/// the tag named in the tagName parameter with the value in the tagValue
/// parameter.
/// </summary>
/// <param name="client">The Amazon EC2 client object used to call
/// the DescribeInstances method.</param>
/// <param name="tagName">A string representing the name of the tag to
/// filter on.</param>
/// <param name="tagValue">A string representing the value of the tag
/// to filter on.</param>
public static async Task GetInstanceDescriptionsFiltered(AmazonEC2Client
client,
    string tagName, string tagValue)
{
    // This is the tag we want to use to filter
    // the results of our list of instances.
    var filters = new List<Filter>
    {
        new Filter
        {
            Name = $"tag:{tagName}",
            Values = new List<string>
            {

```

```
        tagValue,
    },
},
];
var request = new DescribeInstancesRequest
{
    Filters = filters,
};

Console.WriteLine("\nShowing instances with tag: \"IncludeInList\" set
to \"Yes\".");
var paginator = client.Paginator.DescribeInstances(request);

await foreach (var response in paginator.Responses)
{
    foreach (var reservation in response.Reservations)
    {
        foreach (var instance in reservation.Instances)
        {
            Console.Write($"Instance ID: {instance.InstanceId} ");
            Console.WriteLine($"\\tCurrent State:
{instance.State.Name}");
        }
    }
}
}
```

- For API details, see [DescribeInstances](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String describeEC2Instances( Ec2Client ec2, String newInstanceId)
{
    try {
        String pubAddress = "";
        boolean isRunning = false;
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(newInstanceId)
            .build();

        while (!isRunning) {
            DescribeInstancesResponse response =
                ec2.describeInstances(request);
            String state =
                response.reservations().get(0).instances().get(0).state().name();
            if (state.compareTo("RUNNING") == 0) {
                System.out.println("Image id is " +
                    response.reservations().get(0).instances().get(0).imageId());
                System.out.println("Instance type is " +
                    response.reservations().get(0).instances().get(0).instanceType());
                System.out.println("Instance state is " +
                    response.reservations().get(0).instances().get(0).state().name());
                pubAddress =
                    response.reservations().get(0).instances().get(0).publicIpAddress();
                System.out.println("Instance address is " + pubAddress);
            }
        }
    }
}
```

```
        isRunning = true;
    }
}
return pubAddress;
} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [DescribeInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""
    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object that
                         wraps instance actions.
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def display(self, indent=1):
        """
        Displays information about an instance.

        :param indent: The visual indent to apply to the output.
        """
        if self.instance is None:
            logger.info("No instance to display.")
            return

        try:
            self.instance.load()
            ind = '\t'*indent
            print(f"{ind}ID: {self.instance.id}")
            print(f"{ind}Image ID: {self.instance.image_id}")
            print(f"{ind}Instance type: {self.instance.instance_type}")
            print(f"{ind}Key name: {self.instance.key_name}")
            print(f"{ind}VPC ID: {self.instance.vpc_id}")
            print(f"{ind}Public IP: {self.instance.public_ip_address}")
            print(f"{ind}State: {self.instance.state['Name']}")
        except ClientError as err:
            logger.error(
                "Couldn't display your instance. Here's why: %s: %s",
                err.response['Error']['Code'],
                err.response['Error']['Message']
            )
```

```
    err.response['Error']['Code'], err.response['Error']['Message'])
raise
```

- For API details, see [DescribeInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_state(client: &Client, ids: Option<Vec<String>>) -> Result<(), Error> {
    let resp = client
        .describe_instances()
        .set_instance_ids(ids)
        .send()
        .await?;

    for reservation in resp.reservations().unwrap_or_default() {
        for instance in reservation.instances().unwrap_or_default() {
            println!("Instance ID: {}", instance.instance_id().unwrap());
            println!(
                "State:      {:?}",
                instance.state().unwrap().name().unwrap()
            );
            println!();
        }
    }
    Ok(())
}
```

- For API details, see [DescribeInstances](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_ec2->describeinstances( ) .
    "
oo_result is returned for testing purpose "

    " Retrieving details of EC2 instance(s) "
```

```
DATA: lv_instance_id      TYPE /aws1/ec2string,
      lv_status          TYPE /aws1/ec2instancestatename,
      lv_instance_type   TYPE /aws1/ec2instancetype,
      lv_image_id        TYPE /aws1/ec2string.
LOOP AT oo_result->get_reservations( ) INTO DATA(lo_reservation).
  LOOP AT lo_reservation->get_instances( ) INTO DATA(lo_instance).
    lv_instance_id = lo_instance->get_instanceid( ).
    lv_status = lo_instance->get_state( )->get_name( ).
    lv_instance_type = lo_instance->get_instancetype( ).
    lv_image_id = lo_instance->get_imageid( ).
  ENDLOOP.
ENDLOOP.
MESSAGE 'Retrieved information about EC2 instances' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
  DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.
  MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DescribeInstances](#) in *AWS SDK for SAP ABAP API reference*.

## Describe one or more Amazon EBS snapshots using an AWS SDK

The following code example shows how to describe Amazon EBS snapshots.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Shows the state of a snapshot.

```
async fn show_state(client: &Client, id: &str) -> Result<(), Error> {
    let resp = client
        .describe_snapshots()
        .filters(Filter::builder().name("snapshot-id").values(id).build())
        .send()
        .await?;

    println!(
        "State: {}",
        resp.snapshots()
            .unwrap()
            .first()
            .unwrap()
            .state()
            .unwrap()
            .as_ref()
    );
    Ok(())
}
```

```
async fn show_snapshots(client: &Client) -> Result<(), Error> {
    // "self" represents your account ID.
    // You can list the snapshots for any account by replacing
    // "self" with that account ID.
    let resp = client.describe_snapshots().owner_ids("self").send().await?;
    let snapshots = resp.snapshots().unwrap();
    let length = snapshots.len();

    for snapshot in snapshots {
        println!(
            "ID:          {}",
            snapshot.snapshot_id().unwrap_or_default()
        );
        println!(
            "Description: {}",
            snapshot.description().unwrap_or_default()
        );
        println!("State:      {}", snapshot.state().unwrap().as_ref());
        println!();
    }

    println!();
    println!("Found {} snapshot(s)", length);
    println!();

    Ok(())
}
```

- For API details, see [DescribeSnapshots](#) in *AWS SDK for Rust API reference*.

## Disassociate an Elastic IP address from an Amazon EC2 instance using an AWS SDK

The following code examples show how to disassociate an Elastic IP address from an Amazon EC2 instance.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
        DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DisassociateAddress](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ElasticIpWrapper:  
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address  
    actions."""  
    def __init__(self, ec2_resource, elastic_ip=None):  
        """  
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource  
            is used to create additional high-level objects  
            that wrap low-level Amazon EC2 service actions.  
        :param elastic_ip: A Boto3 VpcAddress object. This is a high-level object  
        that  
            wraps Elastic IP actions.  
        """  
        self.ec2_resource = ec2_resource  
        self.elastic_ip = elastic_ip  
  
    @classmethod  
    def from_resource(cls):  
        ec2_resource = boto3.resource('ec2')  
        return cls(ec2_resource)  
  
    def disassociate(self):  
        """  
        Removes an association between an Elastic IP address and an instance. When  
        the  
        association is removed, the instance is assigned a new public IP address.  
        """  
        if self.elastic_ip is None:  
            logger.info("No Elastic IP to disassociate.")  
            return  
  
        try:  
            self.elastic_ip.association.delete()  
        except ClientError as err:  
            logger.error(  
                "Couldn't disassociate Elastic IP %s from its instance. Here's why:  
                %s: %s",  
                self.elastic_ip.allocation_id,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise
```

- For API details, see [DisassociateAddress](#) in *AWS SDK for Python (Boto3) API Reference*.

## Enables monitoring for a running Amazon EC2 instance using an AWS SDK

The following code examples show how to enable monitoring for a running Amazon EC2 instance.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn enable_monitoring(client: &Client, id: &str) -> Result<(), Error> {
    client.monitor_instances().instance_ids(id).send().await?;

    println!("Enabled monitoring");

    Ok(())
}
```

- For API details, see [MonitorInstances](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
  APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
lt_instance_ids.

"Perform Dry Run"
TRY.
  " DryRun is set to true to check if we have the required permissions to
monitor the instance without actually making the request "
  lo_ec2->monitorinstances(
    it_instanceids = lt_instance_ids
    iv_dryrun = abap_true
  ).
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
  " If the error code returned is `DryRunOperation`, it means we have the
required permissions to monitor this instance "
  IF lo_exception->av_err_code = 'DryRunOperation'.
    MESSAGE 'Dry run to enable detailed monitoring completed' TYPE 'I'.
    " DryRun is set to false to enable detailed monitoring "
    lo_ec2->monitorinstances(
      it_instanceids = lt_instance_ids
      iv_dryrun = abap_false
    ).
    MESSAGE 'Detailed monitoring enabled' TYPE 'I'.
    " If the error code returned is `UnauthorizedOperation`, it means we do
not have the required permissions to monitor this instance "
```

```
ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
    MESSAGE 'Dry run to enable detailed monitoring failed: User does not have
the permissions to monitor the instance' TYPE 'E'.
ELSE.
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDIF.
ENDTRY.
```

- For API details, see [MonitorInstances](#) in *AWS SDK for SAP ABAP API reference*.

## Get data about Amazon Machine Images using an AWS SDK

The following code example shows how to get data about Amazon Machine Images (AMIs).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""
    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object that
                         wraps instance actions.
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def get_images(self, image_ids):
        """
        Gets information about Amazon Machine Images (AMIs) from a list of AMI IDs.

        :param image_ids: The list of AMIs to look up.
        :return: A list of Boto3 Image objects that represent the requested AMIs.
        """
        try:
            images = list(self.ec2_resource.images.filter(ImageIds=image_ids))
        except ClientError as err:
            logger.error(
                "Couldn't get images. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return images
```

- For API details, see [DescribeImages](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get data about an Amazon EC2 security group using an AWS SDK

The following code examples show how to get data about an Amazon EC2 security group.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
        .groupIds(groupId)
        .build();

        DescribeSecurityGroupsResponse response =
ec2.describeSecurityGroups(request);
        for(SecurityGroup group : response.securityGroups()) {
            System.out.println( "Found Security Group with Id "
+group.groupId() + " and group VPC "+ group.vpcId());
        }
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for Java 2.x API Reference*.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
actions."""
    def __init__(self, ec2_resource, security_group=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                           is used to create additional high-level objects
                           that wrap low-level Amazon EC2 service actions.
        :param security_group: A Boto3 SecurityGroup object. This is a high-level
                           object
                           that wraps security group actions.
        """
        self.ec2_resource = ec2_resource
        self.security_group = security_group

    @classmethod
```

```
def from_resource(cls):
    ec2_resource = boto3.resource('ec2')
    return cls(ec2_resource)

def describe(self):
    """
    Displays information about the security group.
    """
    if self.security_group is None:
        logger.info("No security group to describe.")
        return

    try:
        print(f"Security group: {self.security_group.group_name}")
        print(f"\tID: {self.security_group.id}")
        print(f"\tVPC: {self.security_group.vpc_id}")
        if self.security_group.ip_permissions:
            print(f"\tInbound permissions:")
            pp(self.security_group.ip_permissions)
    except ClientError as err:
        logger.error(
            "Couldn't get data for security group %s. Here's why: %s: %s",
            self.security_group.id,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  DATA lt_group_ids TYPE /aws1/cl_ec2groupidstrlist_w=>tt_groupidstringlist.
  APPEND NEW /aws1/cl_ec2groupidstrlist_w( iv_value = iv_group_id ) TO
  lt_group_ids.
  oo_result = lo_ec2->describesecuritygroups( it_groupids = lt_group_ids ).
  " oo_result is returned for testing purpose "
  DATA(lt_security_groups) = oo_result->get_securitygroups( ).
  MESSAGE 'Retrieved information about security group(s)' TYPE 'I'.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
  DATA(lv_error) = |" { lo_exception->av_err_code }" - { lo_exception-
  >av_err_msg }|.
  MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for SAP ABAP API reference*.

## Get data about Amazon EC2 instance types using an AWS SDK

The following code examples show how to get data about Amazon EC2 instance types.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType="";
    try {
        List<Filter> filters = new ArrayList<>();
        Filter filter = Filter.builder()
            .name("processor-info.supported-architecture")
            .values("arm64")
            .build();

        filters.add(filter);
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .filters(filters)
            .maxResults(10)
            .build();

        DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
        List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
        for (InstanceTypeInfo type: instanceTypes) {
            System.out.println("The memory information of this type is
"+type.memoryInfo().sizeInMiB());
            System.out.println("Network information is
"+type.networkInfo().toString());
            instanceType = type.instanceType().toString();
        }
    }

    return instanceType;
} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [DescribeInstanceTypes](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""
    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
        """


```

```

        that wrap low-level Amazon EC2 service actions.
:param instance: A Boto3 Instance object. This is a high-level object that
                  wraps instance actions.
"""
self.ec2_resource = ec2_resource
self.instance = instance

@classmethod
def from_resource(cls):
    ec2_resource = boto3.resource('ec2')
    return cls(ec2_resource)

def get_instance_types(self, architecture):
    """
    Gets instance types that support the specified architecture and are
    designated
        as either 'micro' or 'small'. When an instance is created, the instance
    type
        you specify must support the architecture of the AMI you use.

    :param architecture: The kind of architecture the instance types must
    support,
                           such as 'x86_64'.
    :return: A list of instance types that support the specified architecture
            and are either 'micro' or 'small'.
    """
try:
    inst_types = []
    it Paginator =
self.ec2_resource.meta.client.getPaginator('describe_instance_types')
    for page in itPaginator.paginate(
        Filters=[{
            'Name': 'processor-info.supported-architecture', 'Values':
[architecture],
            {'Name': 'instance-type', 'Values': ['*.micro',
'*.*small*']}]):
        inst_types += page['InstanceTypes']
    except ClientError as err:
        logger.error(
            "Couldn't get instance types. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return inst_types

```

- For API details, see [DescribeInstanceTypes](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get details about Elastic IP addresses using an AWS SDK

The following code example shows how to get details about Elastic IP addresses.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_ec2->describeaddresses( ) .  
    "oo_result is returned for testing purpose "  
    DATA(lt_addresses) = oo_result->get_addresses( ).  
    MESSAGE 'Retrieved information about Elastic IP addresses' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.  
        MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [DescribeAddresses](#) in *AWS SDK for SAP ABAP API reference*.

## List Amazon EC2 security key pairs using an AWS SDK

The following code examples show how to list Amazon EC2 security key pairs.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyPairWrapper:  
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions."""  
    def __init__(self, ec2_resource, key_file_dir, key_pair=None):  
        """  
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource  
            is used to create additional high-level objects  
            that wrap low-level Amazon EC2 service actions.  
        :param key_file_dir: The folder where the private key information is  
            stored.  
            This should be a secure folder.  
        :param key_pair: A Boto3 KeyPair object. This is a high-level object that  
            wraps key pair actions.  
        """  
        self.ec2_resource = ec2_resource  
        self.key_pair = key_pair  
        self.key_file_path = None  
        self.key_file_dir = key_file_dir  
  
    @classmethod  
    def from_resource(cls):  
        ec2_resource = boto3.resource('ec2')  
        return cls(ec2_resource, tempfile.TemporaryDirectory())  
  
    def list(self, limit):  
        """  
        Displays a list of key pairs for the current account.  
        :param limit: The maximum number of key pairs to list.  
        """  
        try:  
            for kp in self.ec2_resource.key_pairs.limit(limit):  
                print(f"Found {kp.key_type} key {kp.name} with fingerprint:")  
                print(f"\t{kp.key_fingerprint}")  
        except ClientError as err:  
            logger.error(
```

```
"Couldn't list key pairs. Here's why: %s: %s",
err.response['Error']['Code'], err.response['Error']['Message'])
raise
```

- For API details, see [DescribeKeyPairs](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_ec2->describekeypairs( ) .
    "oo_result is returned for testing purpose "
    DATA(lt_key_pairs) = oo_result->get_keypairs( ).
    MESSAGE 'Retrieved information about key pair(s)' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DescribeKeyPairs](#) in *AWS SDK for SAP ABAP API reference*.

## Reboot an Amazon EC2 instance using an AWS SDK

The following code examples show how to reboot an Amazon EC2 instance.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to reboot Amazon Elastic Compute Cloud (Amazon EC2) instances
/// using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class RebootInstances
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and then calls
```

```
    ///> RebootInstancesAsync to reboot the instance(s) in the ec2InstanceId
    ///> list.
    ///> </summary>
    public static async Task Main()
    {
        string ec2InstanceId = "i-0123456789abcdef0";

        // If your EC2 instances are not in the same AWS Region as
        // the default users on your system, you need to supply
        // the AWS Region as a parameter to the client constructor.
        var client = new AmazonEC2Client();

        var request = new RebootInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

        var response = await client.RebootInstancesAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("Instance(s) successfully rebooted.");
        }
        else
        {
            Console.WriteLine("Could not reboot one or more instances.");
        }
    }
}
```

- For API details, see [RebootInstances](#) in *AWS SDK for .NET API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn reboot_instance(client: &Client, id: &str) -> Result<(), Error> {
    client.reboot_instances().instance_ids(id).send().await?;

    println!("Rebooted instance.");
    Ok(())
}
```

- For API details, see [RebootInstances](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
lt_instance_ids.

"Perform Dry Run"
TRY.
  " DryRun is set to true to check if we have the required permissions to
  reboot the instance without actually making the request "
  lo_ec2->rebootinstances(
    it_instanceids = lt_instance_ids
    iv_dryrun = abap_true
  ).
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    " If the error code returned is 'DryRunOperation', it means we have the
    required permissions to reboot this instance "
    IF lo_exception->av_err_code = 'DryRunOperation'.
      MESSAGE 'Dry run to reboot instance completed' TYPE 'I'.
      " DryRun is set to false to make a reboot request "
      lo_ec2->rebootinstances(
        it_instanceids = lt_instance_ids
        iv_dryrun = abap_false
      ).
      MESSAGE 'Instance rebooted' TYPE 'I'.
      " If the error code returned is 'UnauthorizedOperation', it means we do
      not have the required permissions to reboot this instance "
      ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
        MESSAGE 'Dry run to reboot instance failed: User does not have the
        permission to reboot the instance' TYPE 'E'.
      ELSE.
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
      ENDIF.
    ENDTRY.
```

- For API details, see [RebootInstances](#) in *AWS SDK for SAP ABAP API reference*.

## Release an Elastic IP address using an AWS SDK

The following code examples show how to release an Elastic IP address.

Java

### SDK for Java 2.x

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void releaseEC2Address(Ec2Client ec2, String allocId) {
  try {
    ReleaseAddressRequest request = ReleaseAddressRequest.builder()
```

```
        .allocationId(allocId)
        .build();

        ec2.releaseAddress(request);
        System.out.printf("Successfully released elastic IP address %s",
allocId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ReleaseAddress](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
actions."""
    def __init__(self, ec2_resource, elastic_ip=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param elastic_ip: A Boto3 VpcAddress object. This is a high-level object
                            that
                            wraps Elastic IP actions.
        """
        self.ec2_resource = ec2_resource
        self.elastic_ip = elastic_ip

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def release(self):
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
        it can no longer be used.
        """
        if self.elastic_ip is None:
            logger.info("No Elastic IP to release.")
            return

        try:
            self.elastic_ip.release()
        except ClientError as err:
            logger.error(
                "Couldn't release Elastic IP address %s. Here's why: %s: %s",
                self.elastic_ip.allocation_id,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

- For API details, see [ReleaseAddress](#) in *AWS SDK for Python (Boto3) API Reference*.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_ec2->releaseaddress( iv_allocationid = iv_allocation_id ).  
    MESSAGE 'Elastic IP address released' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
        >av_err_msg }|.  
        MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [ReleaseAddress](#) in *AWS SDK for SAP ABAP API reference*.

## Set inbound rules for an Amazon EC2 security group using an AWS SDK

The following code example shows how to set inbound rules for an Amazon EC2 security group.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecurityGroupWrapper:  
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group  
    actions."""  
    def __init__(self, ec2_resource, security_group=None):  
        """  
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource  
            is used to create additional high-level objects  
            that wrap low-level Amazon EC2 service actions.  
        :param security_group: A Boto3 SecurityGroup object. This is a high-level  
            object  
            that wraps security group actions.  
        """  
        self.ec2_resource = ec2_resource  
        self.security_group = security_group  
  
    @classmethod  
    def from_resource(cls):  
        ec2_resource = boto3.resource('ec2')  
        return cls(ec2_resource)
```

```

def authorize_ingress(self, ssh_ingress_ip):
    """
    Adds a rule to the security group to allow access to SSH.

    :param ssh_ingress_ip: The IP address that is granted inbound access to
    connect
                           to port 22 over TCP, used for SSH.
    :return: The response to the authorization request. The 'Return' field of
    the
                           response indicates whether the request succeeded or failed.
    """
    if self.security_group is None:
        logger.info("No security group to update.")
        return

    try:
        ip_permissions = [
            # SSH ingress open to only the specified IP address.
            'IpProtocol': 'tcp', 'FromPort': 22, 'ToPort': 22,
            'IpRanges': [{'CidrIp': f'{ssh_ingress_ip}/32'}]]
        response =
    self.security_group.authorize_ingress(IpPermissions=ip_permissions)
    except ClientError as err:
        logger.error(
            "Couldn't authorize inbound rules for %s. Here's why: %s: %s",
            self.security_group.id,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response

```

- For API details, see [AuthorizeSecurityGroupIngress](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start an Amazon EC2 instance using an AWS SDK

The following code examples show how to start an Amazon EC2 instance.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to start a list of Amazon Elastic Compute Cloud (Amazon EC2)
/// instances using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class StartInstances
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and uses it to call the
    /// StartInstancesAsync method to start the listed Amazon EC2 instances.

```

```
/// </summary>
public static async Task Main()
{
    string ec2InstanceId = "i-0123456789abcdef0";

    // If your EC2 instances are not in the same AWS Region as
    // the default users on your system, you need to supply
    // the AWS Region as a parameter to the client constructor.
    var client = new AmazonEC2Client();

    var request = new StartInstancesRequest
    {
        InstanceIds = new List<string> { ec2InstanceId },
    };

    var response = await client.StartInstancesAsync(request);

    if (response.StartingInstances.Count > 0)
    {
        var instances = response.StartingInstances;
        instances.ForEach(i =>
        {
            Console.WriteLine($"Successfully started the EC2 Instance with
InstanceID: {i.InstanceId}.");
        });
    }
}
```

- For API details, see [StartInstances](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void startInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to run. This
will take a few minutes.");
    ec2.startInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully started instance "+instanceId);
}
```

- For API details, see [StartInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:  
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""  
    def __init__(self, ec2_resource, instance=None):  
        """  
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource  
                            is used to create additional high-level objects  
                            that wrap low-level Amazon EC2 service actions.  
        :param instance: A Boto3 Instance object. This is a high-level object that  
                         wraps instance actions.  
        """  
        self.ec2_resource = ec2_resource  
        self.instance = instance  
  
    @classmethod  
    def from_resource(cls):  
        ec2_resource = boto3.resource('ec2')  
        return cls(ec2_resource)  
  
    def start(self):  
        """  
        Starts an instance and waits for it to be in a running state.  
        :return: The response to the start request.  
        """  
        if self.instance is None:  
            logger.info("No instance to start.")  
            return  
  
        try:  
            response = self.instance.start()  
            self.instance.wait_until_running()  
        except ClientError as err:  
            logger.error(  
                "Couldn't start instance %s. Here's why: %s: %s", self.instance.id,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return response
```

- For API details, see [StartInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn start_instance(client: &Client, id: &str) -> Result<(), Error> {
    client.start_instances().instance_ids(id).send().await?;

    println!("Started instance.");
    Ok(())
}
```

- For API details, see [StartInstances](#) in *AWS SDK for Rust API reference*.

SAP ABAP

**SDK for SAP ABAP**

**Note**

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
lt_instance_ids.

"Perform Dry Run"
TRY.
    " DryRun is set to true to check if we have the required permissions to
    start the instance without actually making the request "
    lo_ec2->startinstances(
        it_instanceids = lt_instance_ids
        iv_dryrun = abap_true
    ).
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        " If the error code returned is 'DryRunOperation', it means we have the
        required permissions to start this instance "
        IF lo_exception->av_err_code = 'DryRunOperation'.
            MESSAGE 'Dry run to start instance completed' TYPE 'I'.
            " DryRun is set to false to start instance "
            oo_result = lo_ec2->startinstances(          " oo_result is returned for
testing purpose "
                it_instanceids = lt_instance_ids
                iv_dryrun = abap_false
            ).
            MESSAGE 'Successfully started the EC2 instance' TYPE 'I'.
            " If the error code returned is 'UnauthorizedOperation', it means we do
not have the required permissions to start this instance "
            ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
                MESSAGE 'Dry run to start instance failed: User does not have the
permissions to start the instance' TYPE 'E'.
            ELSE.
                DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
            ENDIF.
        ENDIF.
    ENDTRY.
```

```
MESSAGE lv_error TYPE 'E'.
ENDIF.
ENDTRY.
```

- For API details, see [StartInstances](#) in *AWS SDK for SAP ABAP API reference*.

## Stop an Amazon EC2 instance using an AWS SDK

The following code examples show how to stop an Amazon EC2 instance.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

///<summary>
/// Shows how to stop a list of Amazon Elastic Compute Cloud (Amazon EC2)
/// instances using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
///</summary>
public class StopInstances
{
    ///<summary>
    /// Initializes the Amazon EC2 client object and uses it to call the
    /// StartInstancesAsync method to stop the listed Amazon EC2 instances.
    ///</summary>
    public static async Task Main()
    {
        string ec2InstanceId = "i-0123456789abcdef0";

        // If your EC2 instances are not in the same AWS Region as
        // the default user on your system, you need to supply
        // the AWS Region as a parameter to the client constructor.
        var client = new AmazonEC2Client();

        // In addition to the list of instance Ids, the
        // request can also include the following properties:
        // Force      When true forces the instances to
        //             stop but you have to check the integrity
        //             of the file system. Not recommended on
        //             Windows instances.
        // Hibernate  When true, hibernates the instance if the
        //             instance was enabled for hibernation when
        //             it was launched.
        var request = new StopInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

        var response = await client.StopInstancesAsync(request);

        if (response.StoppingInstances.Count > 0)
```

```
        {
            var instances = response.StoppingInstances;
            instances.ForEach(i =>
            {
                Console.WriteLine($"Successfully stopped the EC2 Instance " +
                    $"with InstanceID: {i.InstanceId}.");
            });
        }
    }
```

- For API details, see [StopInstances](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance "+instanceId);
}
```

- For API details, see [StopInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""
    def __init__(self, ec2_resource, instance=None):
```

```
"""
:param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                     is used to create additional high-level objects
                     that wrap low-level Amazon EC2 service actions.
:param instance: A Boto3 Instance object. This is a high-level object that
                  wraps instance actions.
"""
self.ec2_resource = ec2_resource
self.instance = instance

@classmethod
def from_resource(cls):
    ec2_resource = boto3.resource('ec2')
    return cls(ec2_resource)

def stop(self):
    """
    Stops an instance and waits for it to be in a stopped state.

    :return: The response to the stop request.
    """
    if self.instance is None:
        logger.info("No instance to stop.")
        return

    try:
        response = self.instance.stop()
        self.instance.wait_until_stopped()
    except ClientError as err:
        logger.error(
            "Couldn't stop instance %s. Here's why: %s: %s",
            self.instance.id,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return response
```

- For API details, see [StopInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn stop_instance(client: &Client, id: &str) -> Result<(), Error> {
    client.stop_instances().instance_ids(id).send().await?;

    println!("Stopped instance.");

    Ok(())
}
```

- For API details, see [StopInstances](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
lt_instance_ids.

"Perform Dry Run"
TRY.
    " DryRun is set to true to check if we have the required permissions to
stop the instance without actually making the request "
    lo_ec2->stopinstances(
        it_instanceids = lt_instance_ids
        iv_dryrun = abap_true
    ).
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    " If the error code returned is 'DryRunOperation', it means we have the
required permissions to stop this instance "
    IF lo_exception->av_err_code = 'DryRunOperation'.
        MESSAGE 'Dry run to stop instance completed' TYPE 'I'.
        " DryRun is set to false to stop instance "
        oo_result = lo_ec2->stopinstances(          " oo_result is returned for
testing purpose "
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false
        ).
        MESSAGE 'Successfully stopped the EC2 instance' TYPE 'I'.
        " If the error code returned is 'UnauthorizedOperation', it means we do
not have the required permissions to stop this instance "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
            MESSAGE 'Dry run to stop instance failed: User does not have the
permissions to stop the instance' TYPE 'E'.
        ELSE.
            DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
            MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
```

- For API details, see [StopInstances](#) in *AWS SDK for SAP ABAP API reference*.

## Terminate an Amazon EC2 instance using an AWS SDK

The following code examples show how to terminate an Amazon EC2 instance.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void terminateEC2( Ec2Client ec2, String instanceID) {  
  
    try{  
        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()  
            .instanceIds(instanceID)  
            .build();  
  
        TerminateInstancesResponse response = ec2.terminateInstances(ti);  
        List<InstanceStateChange> list = response.terminatingInstances();  
        for (InstanceStateChange sc : list) {  
            System.out.println("The ID of the terminated instance is " +  
sc.instanceId());  
        }  
  
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [TerminateInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:  
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""  
    def __init__(self, ec2_resource, instance=None):  
        """  
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource  
            is used to create additional high-level objects  
            that wrap low-level Amazon EC2 service actions.  
        :param instance: A Boto3 Instance object. This is a high-level object that  
            wraps instance actions.  
        """  
        self.ec2_resource = ec2_resource  
        self.instance = instance  
  
    @classmethod  
    def from_resource(cls):  
        ec2_resource = boto3.resource('ec2')  
        return cls(ec2_resource)  
  
    def terminate(self):  
        """  
        Terminates an instance and waits for it to be in a terminated state.  
        """  
        if self.instance is None:
```

```
logger.info("No instance to terminate.")
return

instance_id = self.instance.id
try:
    self.instance.terminate()
    self.instance.wait_until_terminated()
    self.instance = None
except ClientError as err:
    logging.error(
        "Couldn't terminate instance %s. Here's why: %s: %s",
        instance_id,
        err.response['Error']['Code'],
        err.response['Error']['Message'])
    raise
```

- For API details, see [TerminateInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios for Amazon EC2 using AWS SDKs

The following code examples show how to use Amazon Elastic Compute Cloud with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Get started with Amazon EC2 instances using an AWS SDK \(p. 660\)](#)

## Get started with Amazon EC2 instances using an AWS SDK

The following code examples show how to:

- Create a key pair that is used to secure SSH communication between your computer and an EC2 instance.
- Create a security group that acts as a virtual firewall for your EC2 instances to control incoming and outgoing traffic.
- Find an Amazon Machine Image (AMI) and a compatible instance type.
- Create an instance that is created from the instance type and AMI you select, and is configured to use the security group and key pair created in this example.
- Stop and restart the instance.
- Create an Elastic IP address and associate it as a consistent IP address for your instance.
- Connect to your instance with SSH, using both its public IP address and your Elastic IP address.
- Clean up all of the resources created by this example.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
 * Before running this Java (v2) code example, set up your development environment,  
 * including your credentials.  
 */
```

```
* For more information, see the following documentation topic:  
*  
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
*  
* This Java example performs the following tasks:  
*  
* 1. Creates an RSA key pair and saves the private key data as a .pem file.  
* 2. Lists key pairs.  
* 3. Creates a security group for the default VPC.  
* 4. Displays security group information.  
* 5. Gets a list of Amazon Linux 2 AMIs and selects one.  
* 6. Gets more information about the image.  
* 7. Gets a list of instance types that are compatible with the selected AMI's  
architecture.  
* 8. Creates an instance with the key pair, security group, AMI, and an instance  
type.  
* 9. Displays information about the instance.  
* 10. Stops the instance and waits for it to stop.  
* 11. Starts the instance and waits for it to start.  
* 12. Allocates an Elastic IP address and associates it with the instance.  
* 13. Displays SSH connection info for the instance.  
* 14. Disassociates and deletes the Elastic IP address.  
* 15. Terminates the instance and waits for it to terminate.  
* 16. Deletes the security group.  
* 17. Deletes the key pair.  
*/  
public class EC2Scenario {  
  
    public static final String DASHES = new String(new char[80]).replace("\0",  
"-");  
    public static void main(String[] args) throws InterruptedException {  
  
        final String usage = "\n" +  
            "Usage:\n" +  
            "  <keyName> <fileName> <groupName> <groupDesc> <vpcId>\n\n" +  
            "Where:\n" +  
            "  keyName - A key pair name (for example, TestKeyPair). \n\n" +  
            "  fileName - A file name where the key information is written to. \n  
\n" +  
            "  groupName - The name of the security group. \n\n" +  
            "  groupDesc - The description of the security group. \n\n" +  
            "  vpcId - A VPC ID. You can get this value from the AWS Management  
Console. \n\n" ;  
  
        if (args.length != 5) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String keyName = args[0];  
        String fileName = args[1];  
        String groupName = args[2];  
        String groupDesc = args[3];  
        String vpcId = args[4];  
  
        Region region = Region.US_WEST_2;  
        Ec2Client ec2 = Ec2Client.builder()  
            .region(region)  
            .credentialsProvider(ProfileCredentialsProvider.create())  
            .build();  
  
        SsmClient ssmClient = SsmClient.builder()  
            .region(region)  
            .credentialsProvider(ProfileCredentialsProvider.create())  
            .build();
```

```
System.out.println(DASHES);
System.out.println("Welcome to the Amazon EC2 example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an RSA key pair and save the private key
material as a .pem file.");
createKeyPair(ec2, keyName, fileName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. List key pairs.");
describeKeys(ec2);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Create a security group.");
String groupId = createSecurityGroup(ec2, groupName, groupDesc, vpcId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Display security group info for the newly created
security group.");
describeSecurityGroups(ec2, groupId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a list of Amazon Linux 2 AMIs and selects one
with amzn2 in the name.");
String instanceId = getParaValues(ssmClient);
System.out.println("The instance Id is "+instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Get more information about an amzn2 image.");
String amiValue = describeImage(ec2, instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of instance types.");
String instanceType = getInstanceTypes(ec2);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Create an instance.");
String newInstanceId = runInstance(ec2, instanceType, keyName, groupName,
amiValue );
System.out.println("The instance Id is "+newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Display information about the running instance. ");
String ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i "+fileName +"ec2-user@"+ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Stop the instance and use a waiter.");
stopInstance(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Start the instance and use a waiter.");
startInstance(ec2, newInstanceId);
ipAddress = describeEC2Instances(ec2, newInstanceId);
```

```
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i "+fileName +"ec2-user@"+ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Allocate an Elastic IP address and associate it
with the instance.");
String allocationId = allocateAddress(ec2);
System.out.println("The allocation Id value is "+allocationId);
String associationId = associateAddress(ec2, newInstanceId, allocationId);
System.out.println("The associate Id value is "+associationId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Describe the instance again.");
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i "+fileName +"ec2-user@"+ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Disassociate and release the Elastic IP address.");
disassociateAddress(ec2, associationId);
releaseEC2Address(ec2, allocationId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Terminate the instance and use a waiter.");
terminateEC2(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Delete the security group.");
deleteEC2SecGroup(ec2, groupId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("17. Delete the key.");
deleteKeys(ec2, keyName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("You successfully completed the Amazon EC2 scenario.!");
");

System.out.println(DASHES);
ec2.close();
}

public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id "
+groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateEC2(Ec2Client ec2, String instanceId) {
```

```
try{
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
    ec2.terminateInstances(ti);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceTerminated(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully started instance "+instanceId);
    System.out.println(instanceId +" is terminated!");

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named "+keyPair);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void releaseEC2Address(Ec2Client ec2, String allocId) {
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address
"+allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
    .associationId(associationId)
    .build();
}
```

```
        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String allocateAddress(Ec2Client ec2) {
    try {
        AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
        return allocateResponse.allocationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void startInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to run. This
will take a few minutes.");
    ec2.startInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();
}
```

```
    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully started instance "+instanceId);
}

public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance "+instanceId);
}

public static String describeEC2Instances( Ec2Client ec2, String newInstanceId)
{
    try {
        String pubAddress = "";
        boolean isRunning = false;
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(newInstanceId)
            .build();

        while (!isRunning) {
            DescribeInstancesResponse response =
ec2.describeInstances(request);
            String state =
response.reservations().get(0).instances().get(0).state().name().name();
            if (state.compareTo("RUNNING") ==0) {
                System.out.println("Image id is " +
response.reservations().get(0).instances().get(0).imageId());
                System.out.println("Instance type is " +
response.reservations().get(0).instances().get(0).instanceType());
                System.out.println("Instance state is " +
response.reservations().get(0).instances().get(0).state().name());
                pubAddress =
response.reservations().get(0).instances().get(0).publicIpAddress();
                System.out.println("Instance address is " + pubAddress);
                isRunning = true;
            }
        }
        return pubAddress;
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static String runInstance(Ec2Client ec2, String instanceType, String
keyName, String groupName, String amiId ) {
```

```

        try {
            RunInstancesRequest runRequest = RunInstancesRequest.builder()
                .instanceType(instanceType)
                .keyName(keyName)
                .securityGroups(groupName)
                .maxCount(1)
                .minCount(1)
                .imageId(amiId)
                .build();

            RunInstancesResponse response = ec2.runInstances(runRequest);
            String instanceId = response.instances().get(0).instanceId();
            System.out.println("Successfully started EC2 instance "+instanceId +" based on AMI "+amiId);
            return instanceId;
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Get a list of instance types.
    public static String getInstanceTypes(Ec2Client ec2) {
        String instanceType="";
        try {
            List<Filter> filters = new ArrayList<>();
            Filter filter = Filter.builder()
                .name("processor-info.supported-architecture")
                .values("arm64")
                .build();

            filters.add(filter);
            DescribeInstanceTypesRequest typesRequest =
            DescribeInstanceTypesRequest.builder()
                .filters(filters)
                .maxResults(10)
                .build();

            DescribeInstanceTypesResponse response =
            ec2.describeInstanceTypes(typesRequest);
            List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
            for (InstanceTypeInfo type: instanceTypes) {
                System.out.println("The memory information of this type is "+type.memoryInfo().sizeInMiB());
                System.out.println("Network information is "+type.networkInfo().toString());
                instanceType = type.instanceType().toString();
            }
        }
        return instanceType;
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Display the Description field that corresponds to the instance Id value.
public static String describeImage(Ec2Client ec2, String instanceId) {
    try {
        DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
            .imageIds(instanceId)
            .build();
    }
}

```

```

        DescribeImagesResponse response = ec2.describeImages(imagesRequest);
        System.out.println("The description of the first image is
"+response.images().get(0).description());
        System.out.println("The name of the first image is
"+response.images().get(0).name());

        // Return the image Id value.
        return response.images().get(0).imageId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Get the Id value of an instance with amzn2 in the name.
public static String getParaValues(SsmClient ssmClient) {
    try {
        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
            .build();

        GetParametersByPathIterable responses =
ssmClient.getParametersByPathPaginator(parameterRequest);
        for
(software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse response :
responses) {
            System.out.println("Test "+response.nextToken());
            List<Parameter> parameterList = response.parameters();
            for (Parameter para: parameterList) {
                System.out.println("The name of the para is: "+para.name());
                System.out.println("The type of the para is: "+para.type());
                if (filterName(para.name())) {
                    return para.value();
                }
            }
        }
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "" ;
}

// Return true if the name has amzn2 in it. For example:
// /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-gp2
private static boolean filterName(String name) {
    String[] parts = name.split("/");
    String myValue = parts[4];
    return myValue.contains("amzn2");
}

public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        DescribeSecurityGroupsResponse response =
ec2.describeSecurityGroups(request);
        for(SecurityGroup group : response.securityGroups()) {

```

```
        System.out.println( "Found Security Group with Id "
+group.groupId() +" and group VPC "+ group.vpcId());
    }

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

}

public static String createSecurityGroup( Ec2Client ec2, String groupName,
String groupDesc, String vpcId) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
        .groupName(groupName)
        .description(groupDesc)
        .vpcId(vpcId)
        .build();

        CreateSecurityGroupResponse resp=
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
        .cidrIp("0.0.0.0/0").build();

        IpPermission ipPerm = IpPermission.builder()
        .ipProtocol("tcp")
        .toPort(80)
        .fromPort(80)
        .ipRanges(ipRange)
        .build();

        IpPermission ipPerm2 = IpPermission.builder()
        .ipProtocol("tcp")
        .toPort(22)
        .fromPort(22)
        .ipRanges(ipRange)
        .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(groupName)
        .ipPermissions(ipPerm, ipPerm2)
        .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
"+groupName);
        return resp.groupId();
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void describeKeys( Ec2Client ec2){
    try {
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();
        response.keyPairs().forEach(keyPair -> System.out.printf(
            "Found key pair with name %s " +
            "and fingerprint %s",
            keyPair.keyName(),
            keyPair.keyFingerprint())
    );
}
```

```
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createKeyPair(Ec2Client ec2, String keyName, String
fileName) {
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        CreateKeyPairResponse response = ec2.createKeyPair(request);
        String content = response.keyMaterial();
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
        writer.write(content);
        writer.close();
        System.out.println("Successfully created key pair named "+keyName);

    } catch (Ec2Exception | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [AllocateAddress](#)
  - [AssociateAddress](#)
  - [AuthorizeSecurityGroupIngress](#)
  - [CreateKeyPair](#)
  - [CreateSecurityGroup](#)
  - [DeleteKeyPair](#)
  - [DeleteSecurityGroup](#)
  - [DescribeImages](#)
  - [DescribeInstanceTypes](#)
  - [DescribeInstances](#)
  - [DescribeKeyPairs](#)
  - [DescribeSecurityGroups](#)
  - [DisassociateAddress](#)
  - [ReleaseAddress](#)
  - [RunInstances](#)
  - [StartInstances](#)
  - [StopInstances](#)
  - [TerminateInstances](#)

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
class Ec2InstanceScenario:  
    """Runs an interactive scenario that shows how to get started using EC2  
    instances."""  
    def __init__(self, inst_wrapper, key_wrapper, sg_wrapper, eip_wrapper,  
                 ssm_client):  
        """  
        :param inst_wrapper: An object that wraps instance actions.  
        :param key_wrapper: An object that wraps key pair actions.  
        :param sg_wrapper: An object that wraps security group actions.  
        :param eip_wrapper: An object that wraps Elastic IP actions.  
        :param ssm_client: A Boto3 AWS Systems Manager client.  
        """  
        self.inst_wrapper = inst_wrapper  
        self.key_wrapper = key_wrapper  
        self.sg_wrapper = sg_wrapper  
        self.eip_wrapper = eip_wrapper  
        self.ssm_client = ssm_client  
  
    @demo_func  
    def create_and_list_key_pairs(self):  
        """  
        1. Creates an RSA key pair and saves its private key data as a .pem file in  
        secure  
            temporary storage. The private key data is deleted after the example  
        completes.  
        2. Lists the first five key pairs for the current account.  
        """  
        print("Let's create an RSA key pair that you can be use to securely connect  
        to "  
             "your EC2 instance.")  
        key_name = q.ask("Enter a unique name for your key: ", q.non_empty)  
        self.key_wrapper.create(key_name)  
        print(f"Created a key pair {self.key_wrapper.key_pair.key_name} and saved  
        the "  
             f"private key to {self.key_wrapper.key_file_path}.\\n")  
        if q.ask("Do you want to list some of your key pairs? (y/n) ", q.is_yesno):  
            self.key_wrapper.list(5)  
  
    @demo_func  
    def create_security_group(self):  
        """  
        1. Creates a security group for the default VPC.  
        2. Adds an inbound rule to allow SSH. The SSH rule allows only  
            inbound traffic from the current computer's public IPv4 address.  
        3. Displays information about the security group.  
  
        This function uses 'http://checkip.amazonaws.com' to get the current public  
        IP  
        address of the computer that is running the example. This method works in  
        most  
        cases. However, depending on how your computer connects to the internet,  
        you  
        might have to manually add your public IP address to the security group by  
        using  
            the AWS Management Console.  
        """  
        print("Let's create a security group to manage access to your instance.")  
        sg_name = q.ask("Enter a unique name for your security group: ",  
                       q.non_empty)  
        security_group = self.sg_wrapper.create(  
            sg_name, "Security group for example: get started with instances.")  
        print(f"Created security group {security_group.group_name} in your default  
        """
```

```
f"VPC {security_group.vpc_id}.\n")

ip_response = urllib.request.urlopen('http://checkip.amazonaws.com')
current_ip_address = ip_response.read().decode('utf-8').strip()
print("Let's add a rule to allow SSH only from your current IP address.")
print(f"Your public IP address is {current_ip_address}.")
q.ask("Press Enter to add this rule to your security group.")
response = self.sg_wrapper.authorize_ingress(current_ip_address)
if response['Return']:
    print("Security group rules updated.")
else:
    print("Couldn't update security group rules.")
self.sg_wrapper.describe()

@demo_func
def create_instance(self):
    """
    1. Gets a list of Amazon Linux 2 AMIs from AWS Systems Manager. Specifying
    the
        '/aws/service/ami-amazon-linux-latest' path returns only the latest
    AMIs.
    2. Gets and displays information about the available AMIs and lets you
    select one.
    3. Gets a list of instance types that are compatible with the selected AMI
    and
        lets you select one.
    4. Creates an instance with the previously created key pair and security
    group,
        and the selected AMI and instance type.
    5. Waits for the instance to be running and then displays its information.
    """
    amiPaginator = self.ssm_client.getPaginator('get_parameters_by_path')
    amiOptions = []
    for page in amiPaginator.paginate(Path='/aws/service/ami-amazon-linux-
latest'):
        amiOptions += page['Parameters']
    amzn2Images = self.inst_wrapper.get_images(
        [opt['Value'] for opt in amiOptions if 'amzn2' in opt['Name']])
    print("Let's create an instance from an Amazon Linux 2 AMI. Here are some
options:")
    imageChoice = q.choose(
        "Which one do you want to use? ", [opt.description for opt in
amzn2Images])
    print("Great choice!\n")

    print(f"Here are some instance types that support the "
          f"{{amzn2Images[imageChoice].architecture}} architecture of the
image:")
    instTypes =
    self.inst_wrapper.get_instance_types(amzn2Images[imageChoice].architecture)
    instTypeChoice = q.choose(
        "Which one do you want to use? ", [it['InstanceType'] for it in
instTypes])
    print("Another great choice.\n")

    print("Creating your instance and waiting for it to start...")
    self.inst_wrapper.create(
        amzn2Images[imageChoice],
        instTypes[instTypeChoice]['InstanceType'],
        self.key_wrapper.key_pair,
        [self.sg_wrapper.security_group])
    print(f"Your instance is ready:\n")
    self.inst_wrapper.display()

    print("You can use SSH to connect to your instance.")
```

```
        print("If the connection attempt times out, you might have to manually
update "
            "the SSH ingress rule for your IP address in the AWS Management
Console.")
        self._display_ssh_info()

    def _display_ssh_info(self):
        """
        Displays an SSH connection string that can be used to connect to a running
        instance.
        """
        print("To connect, open another command prompt and run the following
command:")
        if self.eip_wrapper.elastic_ip is None:
            print(f"\tssh -i {self.key_wrapper.key_file_path} "
                  f"ec2-user@{self.inst_wrapper.instance.public_ip_address}")
        else:
            print(f"\tssh -i {self.key_wrapper.key_file_path} "
                  f"ec2-user@{self.eip_wrapper.elastic_ip.public_ip}")
        q.ask("Press Enter when you're ready to continue the demo.")

@demo_func
def associate_elastic_ip(self):
    """
    1. Allocates an Elastic IP address and associates it with the instance.
    2. Displays an SSH connection string that uses the Elastic IP address.
    """
    print("You can allocate an Elastic IP address and associate it with your
instance\n"
          "to keep a consistent IP address even when your instance restarts.")
    elastic_ip = self.eip_wrapper.allocate()
    print(f"Allocated static Elastic IP address: {elastic_ip.public_ip}.")
    self.eip_wrapper.associate(self.inst_wrapper.instance)
    print(f"Associated your Elastic IP with your instance.")
    print("You can now use SSH to connect to your instance by using the Elastic
IP.")
    self._display_ssh_info()

@demo_func
def stop_and_start_instance(self):
    """
    1. Stops the instance and waits for it to stop.
    2. Starts the instance and waits for it to start.
    3. Displays information about the instance.
    4. Displays an SSH connection string. When an Elastic IP address is
associated
        with the instance, the IP address stays consistent when the instance
stops
        and starts.
    """
    print("Let's stop and start your instance to see what changes.")
    print("Stopping your instance and waiting until it's stopped...")
    self.inst_wrapper.stop()
    print("Your instance is stopped. Restarting...")
    self.inst_wrapper.start()
    print("Your instance is running.")
    self.inst_wrapper.display()
    if self.eip_wrapper.elastic_ip is None:
        print("Every time your instance is restarted, its public IP address
changes.")
    else:
        print("Because you have associated an Elastic IP with your instance,
you can \n"
              "connect by using a consistent IP address after the instance
restarts.")
    self._display_ssh_info()
```

```

@demo_func
def cleanup(self):
    """
    1. Disassociate and delete the previously created Elastic IP.
    2. Terminate the previously created instance.
    3. Delete the previously created security group.
    4. Delete the previously created key pair.
    """
    print("Let's clean everything up. This example created these resources:")
    print(f"\tElastic IP: {self.eip_wrapper.elastic_ip.allocation_id}")
    print(f"\tInstance: {self.inst_wrapper.instance.id}")
    print(f"\tSecurity group: {self.sg_wrapper.security_group.id}")
    print(f"\tKey pair: {self.key_wrapper.key_pair.name}")
    if q.ask("Ready to delete these resources? (y/n) ", q.is_yesno):
        self.eip_wrapper.disassociate()
        print("Disassociated the Elastic IP from the instance.")
        self.eip_wrapper.release()
        print("Released the Elastic IP.")
        print("Terminating the instance and waiting for it to terminate...")
        self.inst_wrapper.terminate()
        print("Instance terminated.")
        self.sg_wrapper.delete()
        print("Deleted security group.")
        self.key_wrapper.delete()
        print("Deleted key pair.")

    def run_scenario(self):
        logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

        print('*'*88)
        print("Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started with instances demo.")
        print('*'*88)

        self.create_and_list_key_pairs()
        self.create_security_group()
        self.create_instance()
        self.stop_and_start_instance()
        self.associate_elastic_ip()
        self.stop_and_start_instance()
        self.cleanup()

        print("\nThanks for watching!")
        print('*'*88)

    if __name__ == '__main__':
        try:
            scenario = Ec2InstanceScenario(
                InstanceWrapper.from_resource(), KeyPairWrapper.from_resource(),
                SecurityGroupWrapper.from_resource(), ElasticIpWrapper.from_resource(),
                boto3.client('ssm'))
            scenario.run_scenario()
        except Exception:
            logging.exception("Something went wrong with the demo.")

```

Define a class that wraps key pair actions.

```

class KeyPairWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions."""
    def __init__(self, ec2_resource, key_file_dir, key_pair=None):
        """

```

```

:param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                    is used to create additional high-level objects
                    that wrap low-level Amazon EC2 service actions.
:param key_file_dir: The folder where the private key information is
                     stored.
                     This should be a secure folder.
:param key_pair: A Boto3 KeyPair object. This is a high-level object that
                  wraps key pair actions.
"""
self.ec2_resource = ec2_resource
self.key_pair = key_pair
self.key_file_path = None
self.key_file_dir = key_file_dir

@classmethod
def from_resource(cls):
    ec2_resource = boto3.resource('ec2')
    return cls(ec2_resource, tempfile.TemporaryDirectory())

def create(self, key_name):
    """
    Creates a key pair that can be used to securely connect to an EC2 instance.
    The returned key pair contains private key information that cannot be
    retrieved
    again. The private key data is stored as a .pem file.

    :param key_name: The name of the key pair to create.
    :return: A Boto3 KeyPair object that represents the newly created key pair.
    """
    try:
        self.key_pair = self.ec2_resource.create_key_pair(KeyName=key_name)
        self.key_file_path = os.path.join(self.key_file_dir.name,
                                         f'{self.key_pair.name}.pem')
        with open(self.key_file_path, 'w') as key_file:
            key_file.write(self.key_pair.key_material)
    except ClientError as err:
        logger.error(
            "Couldn't create key %s. Here's why: %s: %s",
            key_name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return self.key_pair

def list(self, limit):
    """
    Displays a list of key pairs for the current account.

    :param limit: The maximum number of key pairs to list.
    """
    try:
        for kp in self.ec2_resource.key_pairs.limit(limit):
            print(f"Found {kp.key_type} key {kp.name} with fingerprint:")
            print(f"\t{kp.key_fingerprint}")
    except ClientError as err:
        logger.error(
            "Couldn't list key pairs. Here's why: %s: %s",
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise

def delete(self):
    """
    Deletes a key pair.

    if self.key_pair is None:
        logger.info("No key pair to delete.")
    return

```

```
key_name = self.key_pair.name
try:
    self.key_pair.delete()
    self.key_pair = None
except ClientError as err:
    logger.error(
        "Couldn't delete key %s. Here's why: %s : %s",
        key_name,
        err.response['Error']['Code'],
        err.response['Error']['Message'])
    raise
```

Define a class that wraps security group actions.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
    actions."""
    def __init__(self, ec2_resource, security_group=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
            is used to create additional high-level objects
            that wrap low-level Amazon EC2 service actions.
        :param security_group: A Boto3 SecurityGroup object. This is a high-level
            object
            that wraps security group actions.
        """
        self.ec2_resource = ec2_resource
        self.security_group = security_group

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def create(self, group_name, group_description):
        """
        Creates a security group in the default virtual private cloud (VPC) of the
        current account.

        :param group_name: The name of the security group to create.
        :param group_description: The description of the security group to create.
        :return: A Boto3 SecurityGroup object that represents the newly created
        security group.
        """
        try:
            self.security_group = self.ec2_resource.create_security_group(
                GroupName=group_name, Description=group_description)
        except ClientError as err:
            logger.error(
                "Couldn't create security group %s. Here's why: %s: %s",
                group_name,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
        else:
            return self.security_group

    def authorize_ingress(self, ssh_ingress_ip):
        """
        Adds a rule to the security group to allow access to SSH.

        :param ssh_ingress_ip: The IP address that is granted inbound access to
        connect
                    to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of
        the
```

```

        response indicates whether the request succeeded or failed.
"""

if self.security_group is None:
    logger.info("No security group to update.")
    return

try:
    ip_permissions = [
        # SSH ingress open to only the specified IP address.
        'IpProtocol': 'tcp', 'FromPort': 22, 'ToPort': 22,
        'IpRanges': [{'CidrIp': f'{ssh_ingress_ip}/32'}]]
    response =
self.security_group.authorize_ingress(IpPermissions=ip_permissions)
except ClientError as err:
    logger.error(
        "Couldn't authorize inbound rules for %. Here's why: %s: %s",
        self.security_group.id,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response

def describe(self):
"""
Displays information about the security group.
"""
if self.security_group is None:
    logger.info("No security group to describe.")
    return

try:
    print(f"Security group: {self.security_group.group_name}")
    print(f"\tID: {self.security_group.id}")
    print(f"\tVPC: {self.security_group.vpc_id}")
    if self.security_group.ip_permissions:
        print(f"\tInbound permissions:")
        pp(self.security_group.ip_permissions)
except ClientError as err:
    logger.error(
        "Couldn't get data for security group %. Here's why: %s: %s",
        self.security_group.id,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise

def delete(self):
"""
Deletes the security group.
"""
if self.security_group is None:
    logger.info("No security group to delete.")

group_id = self.security_group.id
try:
    self.security_group.delete()
except ClientError as err:
    logger.error(
        "Couldn't delete security group %. Here's why: %s: %s", group_id,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise

```

Define a class that wraps instance actions.

```
class InstanceWrapper:
```

```
"""Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""
def __init__(self, ec2_resource, instance=None):
    """
    :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                        is used to create additional high-level objects
                        that wrap low-level Amazon EC2 service actions.
    :param instance: A Boto3 Instance object. This is a high-level object that
                     wraps instance actions.
    """
    self.ec2_resource = ec2_resource
    self.instance = instance

@classmethod
def from_resource(cls):
    ec2_resource = boto3.resource('ec2')
    return cls(ec2_resource)

def create(
    self, image, instance_type, key_pair, security_groups=None):
    """
    Creates a new EC2 instance. The instance starts immediately after
    it is created.

    The instance is created in the default VPC of the current account.

    :param image: A Boto3 Image object that represents an Amazon Machine Image
                  (AMI)
                           that defines attributes of the instance that is created. The
                  AMI
                           defines things like the kind of operating system and the type
                  of
                           storage used by the instance.
    :param instance_type: The type of instance to create, such as 't2.micro'.
                           The instance type defines things like the number of
                  CPUs and
                           the amount of memory.
    :param key_pair: A Boto3 KeyPair or KeyPairInfo object that represents the
                    key
                           pair that is used to secure connections to the instance.
    :param security_groups: A list of Boto3 SecurityGroup objects that
                           represents the
                           security groups that are used to grant access to
                           the
                           instance. When no security groups are specified,
                           the
                           default security group of the VPC is used.
    :return: A Boto3 Instance object that represents the newly created
            instance.
    """
    try:
        instance_params = {
            'ImageId': image.id, 'InstanceType': instance_type, 'KeyName':
key_pair.name
        }
        if security_groups is not None:
            instance_params['SecurityGroupIds'] = [sg.id for sg in
security_groups]
        self.instance = self.ec2_resource.create_instances(**instance_params,
MinCount=1, MaxCount=1)[0]
        self.instance.wait_until_running()
    except ClientError as err:
        logging.error(
            "Couldn't create instance with image %s, instance type %s, and key
%s. "
            "Here's why: %s: %s", image.id, instance_type, key_pair.name,
err.response['Error']['Code'], err.response['Error']['Message'])

```

```
        raise
    else:
        return self.instance

def display(self, indent=1):
    """
    Displays information about an instance.

    :param indent: The visual indent to apply to the output.
    """
    if self.instance is None:
        logger.info("No instance to display.")
        return

    try:
        self.instance.load()
        ind = '\t'*indent
        print(f"{ind}ID: {self.instance.id}")
        print(f"{ind}Image ID: {self.instance.image_id}")
        print(f"{ind}Instance type: {self.instance.instance_type}")
        print(f"{ind}Key name: {self.instance.key_name}")
        print(f"{ind}VPC ID: {self.instance.vpc_id}")
        print(f"{ind}Public IP: {self.instance.public_ip_address}")
        print(f"{ind}State: {self.instance.state['Name']}")
    except ClientError as err:
        logger.error(
            "Couldn't display your instance. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def terminate(self):
    """
    Terminates an instance and waits for it to be in a terminated state.
    """
    if self.instance is None:
        logger.info("No instance to terminate.")
        return

    instance_id = self.instance.id
    try:
        self.instance.terminate()
        self.instance.wait_until_terminated()
        self.instance = None
    except ClientError as err:
        logger.error(
            "Couldn't terminate instance %. Here's why: %s: %s", instance_id,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def start(self):
    """
    Starts an instance and waits for it to be in a running state.

    :return: The response to the start request.
    """
    if self.instance is None:
        logger.info("No instance to start.")
        return

    try:
        response = self.instance.start()
        self.instance.wait_until_running()
    except ClientError as err:
        logger.error(
            "Couldn't start instance %. Here's why: %s: %s", self.instance.id,
            err.response['Error']['Code'], err.response['Error']['Message'])
```

```
        raise
    else:
        return response

def stop(self):
    """
    Stops an instance and waits for it to be in a stopped state.

    :return: The response to the stop request.
    """
    if self.instance is None:
        logger.info("No instance to stop.")
        return

    try:
        response = self.instance.stop()
        self.instance.wait_until_stopped()
    except ClientError as err:
        logger.error(
            "Couldn't stop instance %s. Here's why: %s: %s",
            self.instance.id,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return response

def get_images(self, image_ids):
    """
    Gets information about Amazon Machine Images (AMIs) from a list of AMI IDs.

    :param image_ids: The list of AMIs to look up.
    :return: A list of Boto3 Image objects that represent the requested AMIs.
    """
    try:
        images = list(self.ec2_resource.images.filter(ImageIds=image_ids))
    except ClientError as err:
        logger.error(
            "Couldn't get images. Here's why: %s: %s",
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return images

def get_instance_types(self, architecture):
    """
    Gets instance types that support the specified architecture and are
    designated
        as either 'micro' or 'small'. When an instance is created, the instance
    type
        you specify must support the architecture of the AMI you use.

    :param architecture: The kind of architecture the instance types must
    support,
        such as 'x86_64'.
    :return: A list of instance types that support the specified architecture
        and are either 'micro' or 'small'.
    """
    try:
        inst_types = []
        it Paginator =
self.ec2_resource.meta.client.getPaginator('describe_instance_types')
        for page in itPaginator.paginate(
            Filters=[{
                'Name': 'processor-info.supported-architecture', 'Values':
[architecture],
                {'Name': 'instance-type', 'Values': ['*.micro',
'*.*small']}])
    
```

```
        inst_types += page['InstanceTypes']
    except ClientError as err:
        logger.error(
            "Couldn't get instance types. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return inst_types
```

Define a class that wraps Elastic IP actions.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
    actions."""
    def __init__(self, ec2_resource, elastic_ip=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param elastic_ip: A Boto3 VpcAddress object. This is a high-level object
                           that
                           wraps Elastic IP actions.
        """
        self.ec2_resource = ec2_resource
        self.elastic_ip = elastic_ip

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def allocate(self):
        """
        Allocates an Elastic IP address that can be associated with an Amazon EC2
        instance. By using an Elastic IP address, you can keep the public IP
        address
        constant even when you restart the associated instance.

        :return: The newly created Elastic IP object. By default, the address is
        not
                associated with any instance.
        """
        try:
            response = self.ec2_resource.meta.client.allocate_address(Domain='vpc')
            self.elastic_ip =
                self.ec2_resource.VpcAddress(response['AllocationId'])
        except ClientError as err:
            logger.error(
                "Couldn't allocate Elastic IP. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return self.elastic_ip

    def associate(self, instance):
        """
        Associates an Elastic IP address with an instance. When this association is
        created, the Elastic IP's public IP address is immediately used as the
        public
        IP address of the associated instance.

        :param instance: A Boto3 Instance object. This is a high-level object that
                        wraps
                        Amazon EC2 instance actions.
        """
```

```
:return: A response that contains the ID of the association.  
"""\n    if self.elastic_ip is None:\n        logger.info("No Elastic IP to associate.")\n        return\n\n    try:\n        response = self.elastic_ip.associate(InstanceId=instance.id)\n    except ClientError as err:\n        logger.error(\n            "Couldn't associate Elastic IP %s with instance %s. Here's why: %s:\n%s",\n            self.elastic_ip.allocation_id, instance.id,\n            err.response['Error']['Code'], err.response['Error']['Message'])\n        raise\n    return response\n\n\ndef disassociate(self):\n    """\n        Removes an association between an Elastic IP address and an instance. When\n        the\n        association is removed, the instance is assigned a new public IP address.\n    """\n    if self.elastic_ip is None:\n        logger.info("No Elastic IP to disassociate.")\n        return\n\n    try:\n        self.elastic_ip.association.delete()\n    except ClientError as err:\n        logger.error(\n            "Couldn't disassociate Elastic IP %s from its instance. Here's why:\n%s: %s",\n            self.elastic_ip.allocation_id,\n            err.response['Error']['Code'], err.response['Error']['Message'])\n        raise\n\n\ndef release(self):\n    """\n        Releases an Elastic IP address. After the Elastic IP address is released,\n        it can no longer be used.\n    """\n    if self.elastic_ip is None:\n        logger.info("No Elastic IP to release.")\n        return\n\n    try:\n        self.elastic_ip.release()\n    except ClientError as err:\n        logger.error(\n            "Couldn't release Elastic IP address %s. Here's why: %s: %s",\n            self.elastic_ip.allocation_id,\n            err.response['Error']['Code'], err.response['Error']['Message'])\n        raise
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [AllocateAddress](#)
  - [AssociateAddress](#)
  - [AuthorizeSecurityGroupIngress](#)
  - [CreateKeyPair](#)
  - [CreateSecurityGroup](#)

- [DeleteKeyPair](#)
- [DeleteSecurityGroup](#)
- [DescribeImages](#)
- [DescribeInstanceTypes](#)
- [DescribeInstances](#)
- [DescribeKeyPairs](#)
- [DescribeSecurityGroups](#)
- [DisassociateAddress](#)
- [ReleaseAddress](#)
- [RunInstances](#)
- [StartInstances](#)
- [StopInstances](#)
- [TerminateInstances](#)

## Cross-service examples for Amazon EC2 using AWS SDKs

The following code examples show how to use Amazon Elastic Compute Cloud with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Create a long-lived Amazon EMR cluster and run several steps using an AWS SDK \(p. 683\)](#)
- [Create a short-lived Amazon EMR cluster and run a step using an AWS SDK \(p. 684\)](#)

## Create a long-lived Amazon EMR cluster and run several steps using an AWS SDK

The following code example shows how to create a long-lived Amazon EMR cluster and run several steps.

### Python

#### SDK for Python (Boto3)

Create a long-lived Amazon EMR cluster that uses Apache Spark to query historical Amazon review data from the [Amazon Customer Reviews Dataset](#). Run a job that gets data for top-rated products in specific categories that contain keywords in their product titles. Job results are written to an Amazon Simple Storage Service (Amazon S3) bucket.

- Create an Amazon S3 bucket and upload a job script.
- Create AWS Identity and Access Management (IAM) roles.
- Create Amazon Elastic Compute Cloud (Amazon EC2) security groups.
- Create a long-lived cluster and run several job steps.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon EC2
- Amazon EMR
- IAM

- Amazon S3

## Create a short-lived Amazon EMR cluster and run a step using an AWS SDK

The following code example shows how to create a short-lived Amazon EMR cluster that runs a step and automatically terminates after the step completes.

Python

### SDK for Python (Boto3)

Create a short-lived Amazon EMR cluster that estimates the value of pi using Apache Spark to parallelize a large number of calculations. The job writes output to Amazon EMR logs and to an Amazon Simple Storage Service (Amazon S3) bucket. The cluster terminates itself after completing the job.

- Create an Amazon S3 bucket and upload a job script.
- Create AWS Identity and Access Management (IAM) roles.
- Create Amazon Elastic Compute Cloud (Amazon EC2) security groups.
- Create a short-lived cluster and run a single job step.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon EC2
- Amazon EMR
- IAM
- Amazon S3

## Code examples for Amazon EC2 Auto Scaling using AWS SDKs

The following code examples show how to use Amazon EC2 Auto Scaling with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon EC2 Auto Scaling using AWS SDKs \(p. 685\)](#)
  - [Create an Auto Scaling group using an AWS SDK \(p. 685\)](#)
  - [Delete an Auto Scaling group using an AWS SDK \(p. 689\)](#)
  - [Disable CloudWatch metrics collection for an Auto Scaling group using an AWS SDK \(p. 692\)](#)
  - [Enable CloudWatch metrics collection for an Auto Scaling group using an AWS SDK \(p. 694\)](#)
  - [Get information about Auto Scaling groups using an AWS SDK \(p. 696\)](#)
  - [Get information about Auto Scaling instances using an AWS SDK \(p. 700\)](#)
  - [Get information about Auto Scaling activities using an AWS SDK \(p. 702\)](#)
  - [Set the desired capacity of an Auto Scaling group using an AWS SDK \(p. 705\)](#)
  - [Terminate an instance in an Auto Scaling group using an AWS SDK \(p. 707\)](#)

- Update an Auto Scaling group using an AWS SDK (p. 710)
- Scenarios for Amazon EC2 Auto Scaling using AWS SDKs (p. 713)
  - Manage Auto Scaling groups and instances using an AWS SDK (p. 713)

## Actions for Amazon EC2 Auto Scaling using AWS SDKs

The following code examples show how to use Amazon EC2 Auto Scaling with AWS SDKs. Each example calls an individual service function.

### Examples

- Create an Auto Scaling group using an AWS SDK (p. 685)
- Delete an Auto Scaling group using an AWS SDK (p. 689)
- Disable CloudWatch metrics collection for an Auto Scaling group using an AWS SDK (p. 692)
- Enable CloudWatch metrics collection for an Auto Scaling group using an AWS SDK (p. 694)
- Get information about Auto Scaling groups using an AWS SDK (p. 696)
- Get information about Auto Scaling instances using an AWS SDK (p. 700)
- Get information about Auto Scaling activities using an AWS SDK (p. 702)
- Set the desired capacity of an Auto Scaling group using an AWS SDK (p. 705)
- Terminate an instance in an Auto Scaling group using an AWS SDK (p. 707)
- Update an Auto Scaling group using an AWS SDK (p. 710)

## Create an Auto Scaling group using an AWS SDK

The following code examples show how to create an Auto Scaling group.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Creates a new Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="client">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
/// <param name="groupName">The name to use for the new Auto Scaling
/// group.</param>
/// <param name="launchTemplateName">The name of the Amazon EC2 launch
template
/// to use to create instances in the group.</param>
/// <param name="serviceLinkedRoleARN">The AWS Identity and Access
/// Management (IAM) service-linked role that provides the permissions
/// to use with the Auto Scaling group.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the operation.</returns>
public static async Task<bool> CreateAutoScalingGroup(
    AmazonAutoScalingClient client,
    string groupName,
```

```
        string launchTemplateName,
        string serviceLinkedRoleARN)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var zoneList = new List<string>
        {
            "us-east-2a",
        };

        var request = new CreateAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            AvailabilityZones = zoneList,
            LaunchTemplate = templateSpecification,
            MaxSize = 1,
            MinSize = 1,
            ServiceLinkedRoleARN = serviceLinkedRoleARN,
        };

        var response = await client.CreateAutoScalingGroupAsync(request);
        Console.WriteLine(groupName + " Auto Scaling Group created");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

- For API details, see [CreateAutoScalingGroup](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
                                         String groupName,
                                         String launchTemplateName,
                                         String serviceLinkedRoleARN,
                                         String vpcZoneId) {

    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
        LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        CreateAutoScalingGroupRequest request =
        CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .availabilityZones("us-east-1a")
            .launchTemplate(templateSpecification)
            .maxSize(1)
            .minSize(1)
            .vpcZoneIdentifier(vpcZoneId)
            .serviceLinkedRoleARN(serviceLinkedRoleARN)
            .build();
    }
}
```

```
        autoScalingClient.createAutoScalingGroup(request);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter.waitUntilGroupExists(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateAutoScalingGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createAutoScalingGroup(groupName: String, launchTemplateNameVal:
String, serviceLinkedRoleARNVal: String, vpcZoneIdVal: String) {
    val templateSpecification = LaunchTemplateSpecification {
        launchTemplateName = launchTemplateNameVal
    }

    val request = CreateAutoScalingGroupRequest {
        autoScalingGroupName = groupName
        availabilityZones = listOf("us-east-1a")
        launchTemplate = templateSpecification
        maxSize = 1
        minSize = 1
        vpcZoneIdentifier = vpcZoneIdVal
        serviceLinkedRoleArn = serviceLinkedRoleARNVal
    }

    // This object is required for the waiter call.
    val groupsRequestWaiter = DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}
```

- For API details, see [CreateAutoScalingGroup](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:  
    """Encapsulates Amazon EC2 Auto Scaling actions."""  
    def __init__(self, autoscaling_client):  
        """  
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.  
        """  
        self.autoscaling_client = autoscaling_client  
  
    def create_group(  
        self, group_name, group_zones, launch_template_name, min_size,  
        max_size):  
        """  
        Creates an Auto Scaling group.  
  
        :param group_name: The name to give to the group.  
        :param group_zones: The Availability Zones in which instances can be  
        created.  
        :param launch_template_name: The name of an existing Amazon EC2 launch  
        template.  
                           The launch template specifies the  
        configuration of  
                           instances that are created by auto scaling  
        activities.  
        :param min_size: The minimum number of active instances in the group.  
        :param max_size: The maximum number of active instances in the group.  
        """  
        try:  
            self.autoscaling_client.create_auto_scaling_group(  
                AutoScalingGroupName=group_name,  
                AvailabilityZones=group_zones,  
                LaunchTemplate={  
                    'LaunchTemplateName': launch_template_name, 'Version':  
                    '$Default'},  
                MinSize=min_size, MaxSize=max_size  
            )  
        except ClientError as err:  
            logger.error(  
                "Couldn't create group %s. Here's why: %s: %s", group_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise
```

- For API details, see [CreateAutoScalingGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn create_group(client: &Client, name: &str, id: &str) -> Result<(), Error> {
    client
        .create_auto_scaling_group()
        .auto_scaling_group_name(name)
        .instance_id(id)
        .min_size(1)
        .max_size(5)
        .send()
        .await?;

    println!("Created AutoScaling group");

    Ok(())
}
```

- For API details, see [CreateAutoScalingGroup](#) in *AWS SDK for Rust API reference*.

## Delete an Auto Scaling group using an AWS SDK

The following code examples show how to delete an Auto Scaling group.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Deletes an Auto Scaling group.
///</summary>
///<param name="autoScalingClient">The initialized Amazon EC2 Auto
Scaling
///<param name="client object.</param>
///<param name="groupName">The name of the Auto Scaling group.</param>
///<returns>A Boolean value that indicates the success or failure of
///the operation.</returns>
public static async Task<bool> DeleteAutoScalingGroupAsync(
    AmazonAutoScalingClient autoScalingClient,
    string groupName)
{
    var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
    {
        AutoScalingGroupName = groupName,
        ForceDelete = true,
    };

    var response = await
autoScalingClient.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"You successfully deleted {groupName}");
        return true;
    }

    Console.WriteLine($"Couldn't delete {groupName}.");
    return false;
}
```

```
}
```

- For API details, see [DeleteAutoScalingGroup](#) in *AWS SDK for .NET API Reference*.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build() ;

        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest) ;
        System.out.println("You successfully deleted "+groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteAutoScalingGroup](#) in *AWS SDK for Java 2.x API Reference*.

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest = DeleteAutoScalingGroupRequest {
        autoScalingGroupName = groupName
        forceDelete = true
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- For API details, see [DeleteAutoScalingGroup](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:  
    """Encapsulates Amazon EC2 Auto Scaling actions."""  
    def __init__(self, autoscaling_client):  
        """  
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.  
        """  
        self.autoscaling_client = autoscaling_client  
  
    def delete_group(self, group_name):  
        """  
        Deletes an Auto Scaling group. All instances must be stopped before the  
        group can be deleted.  
  
        :param group_name: The name of the group to delete.  
        """  
        try:  
            self.autoscaling_client.delete_auto_scaling_group(  
                AutoScalingGroupName=group_name)  
        except ClientError as err:  
            logger.error(  
                "Couldn't delete group %s. Here's why: %s: %s", group_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise
```

- For API details, see [DeleteAutoScalingGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn delete_group(client: &Client, name: &str, force: bool) -> Result<(),  
Error> {  
    client  
        .delete_auto_scaling_group()  
        .auto_scaling_group_name(name)  
        .set_force_delete(if force { Some(true) } else { None })  
        .send()  
        .await?;
```

```
    println!("Deleted Auto Scaling group");
    Ok(())
}
```

- For API details, see [DeleteAutoScalingGroup](#) in *AWS SDK for Rust API reference*.

## Disable CloudWatch metrics collection for an Auto Scaling group using an AWS SDK

The following code examples show how to disable CloudWatch metrics collection for an Auto Scaling group.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Disables the collection of metric data for an Auto Scaling group.
/// </summary>
/// <param name="client">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the operation.</returns>
public static async Task<bool>
DisableMetricsCollectionAsync(AmazonAutoScalingClient client, string groupName)
{
    var request = new DisableMetricsCollectionRequest
    {
        AutoScalingGroupName = groupName,
    };

    var response = await client.DisableMetricsCollectionAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [DisableMetricsCollection](#) in *AWS SDK for .NET API Reference*.

### Java

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
    try {
```

```
DisableMetricsCollectionRequest disableMetricsCollectionRequest =  
    DisableMetricsCollectionRequest.builder()  
        .autoScalingGroupName(groupName)  
        .metrics("GroupMaxSize")  
        .build();  
  
autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);  
System.out.println("The disable metrics collection operation was  
successful");  
  
} catch (AutoScalingException e) {  
    System.err.println(e.awsErrorDetails().errorMessage());  
    System.exit(1);  
}  
}
```

- For API details, see [DisableMetricsCollection](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun disableMetricsCollection(groupName: String) {  
    val disableMetricsCollectionRequest = DisableMetricsCollectionRequest {  
        autoScalingGroupName = groupName  
        metrics = listOf("GroupMaxSize")  
    }  
  
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->  
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)  
        println("The disable metrics collection operation was successful")  
    }  
}
```

- For API details, see [DisableMetricsCollection](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:  
    """Encapsulates Amazon EC2 Auto Scaling actions."""  
    def __init__(self, autoscaling_client):  
        """  
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.  
    
```

```
"""
    self.autoscaling_client = autoscaling_client

def disable_metrics(self, group_name):
    """
    Stops CloudWatch metric collection for the Auto Scaling group.

    :param group_name: The name of the group.
    """
    try:
        self.autoscaling_client.disable_metrics_collection(
            AutoScalingGroupName=group_name)
    except ClientError as err:
        logger.error(
            "Couldn't disable metrics %s. Here's why: %s: %s",
            group_name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
```

- For API details, see [DisableMetricsCollection](#) in *AWS SDK for Python (Boto3) API Reference*.

## Enable CloudWatch metrics collection for an Auto Scaling group using an AWS SDK

The following code examples show how to enable CloudWatch metrics collection for an Auto Scaling group.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Enables the collection of metric data for an Auto Scaling group.
///</summary>
///<param name="client">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
///<param name="groupName">The name of the Auto Scaling group.</param>
///<returns>A Boolean value that indicates the success or failure of
/// the operation.</returns>
public static async Task<bool>
EnableMetricsCollectionAsync(AmazonAutoScalingClient client, string groupName)
{
    var listMetrics = new List<string>
    {
        "GroupMaxSize",
    };

    var collectionRequest = new EnableMetricsCollectionRequest
    {
        AutoScalingGroupName = groupName,
        Metrics = listMetrics,
        Granularity = "1Minute",
    };

    var response = await
    client.EnableMetricsCollectionAsync(collectionRequest);
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- For API details, see [EnableMetricsCollection](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {

        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [EnableMetricsCollection](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest = EnableMetricsCollectionRequest {
        autoScalingGroupName = groupName
        metrics = listOf("GroupMaxSize")
        granularity = "1Minute"
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
    }
}
```

```
        }           println("The enable metrics collection operation was successful")
    }
```

- For API details, see [EnableMetricsCollection](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def enable_metrics(self, group_name, metrics):
        """
        Enables CloudWatch metric collection for Amazon EC2 Auto Scaling
        activities.

        :param group_name: The name of the group to enable.
        :param metrics: A list of metrics to collect.
        """
        try:
            self.autoscaling_client.enable_metrics_collection(
                AutoScalingGroupName=group_name, Metrics=metrics,
                Granularity='1Minute')
        except ClientError as err:
            logger.error(
                "Couldn't enable metrics on %s. Here's why: %s: %s",
                group_name, err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
```

- For API details, see [EnableMetricsCollection](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get information about Auto Scaling groups using an AWS SDK

The following code examples show how to get information about Auto Scaling groups.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Gets data about the instances in an Amazon EC2 Auto Scaling group.
```

```
/// </summary>
/// <param name="client">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A list of Auto Scaling details.</returns>
public static async Task<List<AutoScalingInstanceDetails>>
DescribeAutoScalingInstancesAsync(
    AmazonAutoScalingClient client,
    string groupName)
{
    var groups = await DescribeAutoScalingGroupsAsync(client, groupName);
    var instanceIds = new List<string>();
    groups.ForEach(group =>
    {
        if (group.AutoScalingGroupName == groupName)
        {
            group.Instances.ForEach(instance =>
            {
                instanceIds.Add(instance.InstanceId);
            });
        }
    });

    var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
    {
        MaxRecords = 10,
        InstanceIds = instanceIds,
    };

    var response = await
client.DescribeAutoScalingInstancesAsync(scalingGroupsRequest);
    var instanceDetails = response.AutoScalingInstances;

    return instanceDetails;
}
```

- For API details, see [DescribeAutoScalingGroups](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String getAutoScaling( AutoScalingClient autoScalingClient,
String groupName) {
    try{
        String instanceId = "";
        DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group: groups) {
            System.out.println("The group name is " +
group.autoScalingGroupName());
```

```
        System.out.println("The group ARN is " +  
group.autoScalingGroupARN());  
  
        List<Instance> instances = group.instances();  
        for (Instance instance : instances) {  
            instanceId = instance.instanceId();  
        }  
    }  
    return instanceId;  
} catch (AutoScalingException e) {  
    System.err.println(e.awsErrorDetails().errorMessage());  
    System.exit(1);  
}  
return "";  
}
```

- For API details, see [DescribeAutoScalingGroups](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getAutoScalingGroups(groupName: String) {  
    val scalingGroupsRequest = DescribeAutoScalingGroupsRequest {  
        autoScalingGroupNames = listOf(groupName)  
    }  
  
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->  
        val response =  
        autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)  
        response.autoScalingGroups?.forEach { group ->  
            println("The group name is ${group.autoScalingGroupName}")  
            println("The group ARN is ${group.autoScalingGroupArn}")  
            group.instances?.forEach { instance ->  
                println("The instance id is ${instance.instanceId}")  
                println("The lifecycle state is " + instance.lifecycleState)  
            }  
        }  
    }  
}
```

- For API details, see [DescribeAutoScalingGroups](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:  
    """Encapsulates Amazon EC2 Auto Scaling actions."""  
    def __init__(self, autoscaling_client):  
        """  
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.  
        """  
        self.autoscaling_client = autoscaling_client  
  
    def describe_group(self, group_name):  
        """  
        Gets information about an Auto Scaling group.  
  
        :param group_name: The name of the group to look up.  
        :return: Information about the group, if found.  
        """  
        try:  
            response = self.autoscaling_client.describe_auto_scaling_groups(  
                AutoScalingGroupNames=[group_name])  
        except ClientError as err:  
            logger.error(  
                "Couldn't describe group %s. Here's why: %s: %s", group_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            groups = response.get('AutoScalingGroups', [])  
            return groups[0] if len(groups) > 0 else None
```

- For API details, see [DescribeAutoScalingGroups](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn list_groups(client: &Client) -> Result<(), Error> {  
    let resp = client.describe_auto_scaling_groups().send().await?;  
  
    println!("Groups:");  
  
    let groups = resp.auto_scaling_groups().unwrap_or_default();  
  
    for group in groups {  
        println!("  {}", group.auto_scaling_group_name().unwrap_or_default());  
        println!("  
          ARN:      {}"  
              , group.auto_scaling_group_arn().unwrap_or_default());  
        println!();  
        println!("  Minimum size: {}", group.min_size().unwrap_or_default());  
        println!("  Maximum size: {}", group.max_size().unwrap_or_default());  
        println!();  
    }  
  
    println!("Found {} group(s)", groups.len());
```

```
        Ok(())
    }
```

- For API details, see [DescribeAutoScalingGroups](#) in *AWS SDK for Rust API reference*.

## Get information about Auto Scaling instances using an AWS SDK

The following code examples show how to get information about Auto Scaling instances.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Gets data about the instances in an Amazon EC2 Auto Scaling group.
///</summary>
///<param name="client">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
///<param name="groupName">The name of the Auto Scaling group.</param>
///<returns>A list of Auto Scaling details.</returns>
public static async Task<List<AutoScalingInstanceDetails>>
DescribeAutoScalingInstancesAsync(
    AmazonAutoScalingClient client,
    string groupName)
{
    var groups = await DescribeAutoScalingGroupsAsync(client, groupName);
    var instanceIds = new List<string>();
    groups.ForEach(group =>
    {
        if (group.AutoScalingGroupName == groupName)
        {
            group.Instances.ForEach(instance =>
            {
                instanceIds.Add(instance.InstanceId);
            });
        }
    });

    var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
    {
        MaxRecords = 10,
        InstanceIds = instanceIds,
    };

    var response = await
client.DescribeAutoScalingInstancesAsync(scalingGroupsRequest);
    var instanceDetails = response.AutoScalingInstances;

    return instanceDetails;
}
```

- For API details, see [DescribeAutoScalingInstances](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeAutoScalingInstance( AutoScalingClient autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest =
            = DescribeAutoScalingInstancesRequest.builder()
                .instanceIds(id)
                .build();

        DescribeAutoScalingInstancesResponse response =
            autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
            response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance:instances ) {
            System.out.println("The instance lifecycle state is:
"+instance.lifecycleState());
        }
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeAutoScalingInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest {
        instanceIds = listOf(id)
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group -
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}
```

- For API details, see [DescribeAutoScalingInstances](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:  
    """Encapsulates Amazon EC2 Auto Scaling actions."""  
    def __init__(self, autoscaling_client):  
        """  
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.  
        """  
        self.autoscaling_client = autoscaling_client  
  
    def describe_instances(self, instance_ids):  
        """  
        Gets information about instances.  
  
        :param instance_ids: A list of instance IDs to look up.  
        :return: Information about instances, or an empty list if none are found.  
        """  
        try:  
            response = self.autoscaling_client.describe_auto_scaling_instances(  
                InstanceIds=instance_ids)  
        except ClientError as err:  
            logger.error(  
                "Couldn't describe instances %s. Here's why: %s: %s", instance_ids,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return response['AutoScalingInstances']
```

- For API details, see [DescribeAutoScalingInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get information about Auto Scaling activities using an AWS SDK

The following code examples show how to get information about Auto Scaling activities.

### .NET

#### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Retrieves a list of the Auto Scaling activities for an Auto Scaling  
group.  
/// </summary>  
/// <param name="client">The initialized Amazon EC2 Auto Scaling
```

```
    ///<param name="groupName">The name of the Auto Scaling group.</param>
    ///<returns>A list of Auto Scaling activities.</returns>
    public static async Task<List<Activity>>
DescribeAutoScalingActivitiesAsync(
    AmazonAutoScalingClient client,
    string groupName)
{
    var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
    {
        AutoScalingGroupName = groupName,
        MaxRecords = 10,
    };

    var response = await
client.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
    return response.Activities;
}
```

- For API details, see [DescribeScalingActivities](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
        .autoScalingGroupName(groupName)
        .maxRecords(10)
        .build();

        DescribeScalingActivitiesResponse response =
autoScalingClient.describeScalingActivities(scalingActivitiesRequest);
        List<Activity> activities = response.activities();
        for (Activity activity: activities) {
            System.out.println("The activity Id is "+activity.activityId());
            System.out.println("The activity details are "+activity.details());
        }
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeScalingActivities](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques = DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
        maxRecords = 10
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
${group.healthCheckType}")
        }
    }
}
```

- For API details, see [DescribeScalingActivities](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def describe_scaling_activities(self, group_name):
        """
        Gets information about scaling activities for the group. Scaling activities
        are things like instances stopping or starting in response to user requests
        or capacity changes.

        :param group_name: The name of the group to look up.
        :return: The list of scaling activities for the group, ordered with the
        most
                    recent activity first.
        """
        try:
            response = self.autoscaling_client.describe_scaling_activities(
                AutoScalingGroupName=group_name)
```

```
        except ClientError as err:
            logger.error(
                "Couldn't describe scaling activities %s. Here's why: %s: %s",
                group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['Activities']
```

- For API details, see [DescribeScalingActivities](#) in *AWS SDK for Python (Boto3) API Reference*.

## Set the desired capacity of an Auto Scaling group using an AWS SDK

The following code examples show how to set the desired capacity of an Auto Scaling group.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Sets the desired capacity of an Auto Scaling group.
/// </summary>
/// <param name="client">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <param name="desiredCapacity">The desired capacity for the Auto
/// Scaling group.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the operation.</returns>
public static async Task<bool> SetDesiredCapacityAsync(
    AmazonAutoScalingClient client,
    string groupName,
    int desiredCapacity)
{
    var capacityRequest = new SetDesiredCapacityRequest
    {
        AutoScalingGroupName = groupName,
        DesiredCapacity = desiredCapacity,
    };

    var response = await client.SetDesiredCapacityAsync(capacityRequest);
    Console.WriteLine($"You have set the DesiredCapacity to
{desiredCapacity}.");
}

return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [SetDesiredCapacity](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
        .autoScalingGroupName(groupName)
        .desiredCapacity(2)
        .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SetDesiredCapacity](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest = SetDesiredCapacityRequest {
        autoScalingGroupName = groupName
        desiredCapacity = 2
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}
```

- For API details, see [SetDesiredCapacity](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:  
    """Encapsulates Amazon EC2 Auto Scaling actions."""  
    def __init__(self, autoscaling_client):  
        """  
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.  
        """  
        self.autoscaling_client = autoscaling_client  
  
    def set_desired_capacity(self, group_name, capacity):  
        """  
        Sets the desired capacity of the group. Amazon EC2 Auto Scaling tries to  
        keep the  
        number of running instances equal to the desired capacity.  
  
        :param group_name: The name of the group to update.  
        :param capacity: The desired number of running instances.  
        """  
        try:  
            self.autoscaling_client.set_desired_capacity(  
                AutoScalingGroupName=group_name, DesiredCapacity=capacity,  
                HonorCooldown=False)  
        except ClientError as err:  
            logger.error(  
                "Couldn't set desired capacity %s. Here's why: %s: %s", group_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise
```

- For API details, see [SetDesiredCapacity](#) in *AWS SDK for Python (Boto3) API Reference*.

## Terminate an instance in an Auto Scaling group using an AWS SDK

The following code examples show how to terminate an instance in an Auto Scaling group.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Terminates all instances in the Auto Scaling group in preparation for  
/// deleting the group.  
/// </summary>  
/// <param name="client">The initialized Amazon EC2 Auto Scaling  
/// client object.</param>
```

```
    /// <param name="instanceId">The instance Id of the instance to
    terminate.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public static async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        AmazonAutoScalingClient client,
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
client.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance
{instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }
}
```

- For API details, see [TerminateInstanceInAutoScalingGroup](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId){
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance "+instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [TerminateInstanceInAutoScalingGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request = TerminateInstanceInAutoScalingGroupRequest {
        instanceId = instanceIdVal
        shouldDecrementDesiredCapacity = false
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- For API details, see [TerminateInstanceInAutoScalingGroup](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def terminate_instance(self, instance_id, decrease_capacity):
        """
        Stops an instance.

        :param instance_id: The ID of the instance to stop.
        :param decrease_capacity: Specifies whether to decrease the desired
            capacity
            parameter,
            replacement
            threshold is
            crossed.
        :return: The scaling activity that occurs in response to this action.
        """
        try:
            response =
                self.autoscaling_client.terminate_instance_in_auto_scaling_group(
```

```
InstanceId=instance_id,
ShouldDecrementDesiredCapacity=decrease_capacity)
except ClientError as err:
    logger.error(
        "Couldn't terminate instance %s. Here's why: %s: %s",
        instance_id,
        err.response['Error']['Code'],
        err.response['Error']['Message'])
    raise
else:
    return response['Activity']
```

- For API details, see [TerminateInstanceInAutoScalingGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update an Auto Scaling group using an AWS SDK

The following code examples show how to update the configuration for an Auto Scaling group.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Updates the capacity of an Auto Scaling group.
///</summary>
///<param name="client">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
///<param name="groupName">The name of the Auto Scaling group.</param>
///<param name="launchTemplateName">The name of the EC2 launch template.</param>
///<param name="serviceLinkedRoleARN">The Amazon Resource Name (ARN)
/// of the AWS Identity and Access Management (IAM) service-linked role.</param>
///<param name="maxSize">The maximum number of instances that can be
/// created for the Auto Scaling group.</param>
///<returns>A Boolean value that indicates the success or failure of
/// the update operation.</returns>
public static async Task<bool> UpdateAutoScalingGroupAsync(
    AmazonAutoScalingClient client,
    string groupName,
    string launchTemplateName,
    string serviceLinkedRoleARN,
    int maxSize)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };

    var groupRequest = new UpdateAutoScalingGroupRequest
    {
        MaxSize = maxSize,
        ServiceLinkedRoleARN = serviceLinkedRoleARN,
        AutoScalingGroupName = groupName,
        LaunchTemplate = templateSpecification,
    };
}
```

```
        var response = await client.UpdateAutoScalingGroupAsync(groupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully updated the Auto Scaling group
{groupName}.");
            return true;
        }
        else
        {
            return false;
        }
    }
```

- For API details, see [UpdateAutoScalingGroup](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName, String launchTemplateName, String serviceLinkedRoleARN) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .serviceLinkedRoleARN(serviceLinkedRoleARN)
            .autoScalingGroupName(groupName)
            .launchTemplate(templateSpecification)
            .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter.waitUntilGroupInService(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("You successfully updated the auto scaling group
"+groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [UpdateAutoScalingGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun updateAutoScalingGroup(groupName: String, launchTemplateNameVal: String, serviceLinkedRoleARNVal: String) {
    val templateSpecification = LaunchTemplateSpecification {
        launchTemplateName = launchTemplateNameVal
    }

    val groupRequest = UpdateAutoScalingGroupRequest {
        maxSize = 3
        serviceLinkedRoleArn = serviceLinkedRoleARNVal
        autoScalingGroupName = groupName
        launchTemplate = templateSpecification
    }

    val groupsRequestWaiter = DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group $groupName")
    }
}
```

- For API details, see [UpdateAutoScalingGroup](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def update_group(self, group_name, **kwargs):
        """
        Updates an Auto Scaling group.

        :param group_name: The name of the group to update.
        """


```

```
:param kwargs: Keyword arguments to pass through to the service.  
"""  
try:  
    self.autoscaling_client.update_auto_scaling_group(  
        AutoScalingGroupName=group_name, **kwargs)  
except ClientError as err:  
    logger.error(  
        "Couldn't update group %s. Here's why: %s: %s", group_name,  
        err.response['Error']['Code'], err.response['Error']['Message'])  
    raise
```

- For API details, see [UpdateAutoScalingGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn update_group(client: &Client, name: &str, size: i32) -> Result<(), Error> {  
    client  
        .update_auto_scaling_group()  
        .auto_scaling_group_name(name)  
        .max_size(size)  
        .send()  
        .await?  
  
    println!("Updated AutoScaling group");  
  
    Ok(())
}
```

- For API details, see [UpdateAutoScalingGroup](#) in *AWS SDK for Rust API reference*.

## Scenarios for Amazon EC2 Auto Scaling using AWS SDKs

The following code examples show how to use Amazon EC2 Auto Scaling with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Manage Auto Scaling groups and instances using an AWS SDK \(p. 713\)](#)

## Manage Auto Scaling groups and instances using an AWS SDK

The following code examples show how to:

- Create an Amazon EC2 Auto Scaling group and configure it with a launch template and Availability Zones.
- Get information about the group and running instances.
- Enable Amazon CloudWatch metrics collection on the group.
- Update the desired capacity of the group and wait for an instance to start.
- Terminate an instance in the group.
- List scaling activities that occur in response to user requests and capacity changes.
- Get statistics for CloudWatch metrics that are collected during the example.
- Stop collecting metrics, terminate all instances, and delete the group.

.NET

**AWS SDK for .NET**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
global using Amazon;
global using Amazon.AutoScaling;
global using Amazon.AutoScaling.Model;
global using AutoScale_Basics;

var imageId = "ami-05803413c51f242b7";
var instanceType = "t2.micro";
var launchTemplateName = "AutoScaleLaunchTemplate";

// The name of the Auto Scaling group.
var groupName = "AutoScaleExampleGroup";

// The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM)
// service-linked role.
var serviceLinkedRoleARN = "<Enter Value>";

var client = new AmazonAutoScalingClient(RegionEndpoint.USEast2);

Console.WriteLine("Auto Scaling Basics");
DisplayDescription();

// Create the launch template and save the template Id to use when deleting the
// launch template at the end of the application.
var launchTemplateId = await EC2Methods.CreateLaunchTemplateAsync(imageId,
    instanceType, launchTemplateName);

// Confirm that the template was created by asking for a description of it.
await EC2Methods.DescribeLaunchTemplateAsync(launchTemplateName);

PressEnter();

Console.WriteLine($"---- Creating an Auto Scaling group named {groupName}. ---");
var success = await AutoScaleMethods.CreateAutoScalingGroup(
    client,
    groupName,
    launchTemplateName,
    serviceLinkedRoleARN);

// Keep checking the details of the new group until its lifecycle state
// is "InService".
Console.WriteLine($"Waiting for the Auto Scaling group to be active.");
```

```
List<AutoScalingInstanceDetails> instanceDetails;

do
{
    instanceDetails = await
    AutoScaleMethods.DescribeAutoScalingInstancesAsync(client, groupName);
}
while (instanceDetails.Count <= 0);

Console.WriteLine($"Auto scaling group {groupName} successfully created.");
Console.WriteLine($"{instanceDetails.Count} instances were created for the
group.");

// Display the details of the Auto Scaling group.
instanceDetails.ForEach(detail =>
{
    Console.WriteLine($"Group name: {detail.AutoScalingGroupName}");
});

PressEnter();

Console.WriteLine($"\\n--- Enable metrics collection for {groupName}");
await AutoScaleMethods.EnableMetricsCollectionAsync(client, groupName);

// Show the metrics that are collected for the group.

// Update the maximum size of the group to three instances.
Console.WriteLine(" --- Update the Auto Scaling group to increase max size to 3
---");
int maxSize = 3;
await AutoScaleMethods.UpdateAutoScalingGroupAsync(client, groupName,
launchTemplateName, serviceLinkedRoleARN, maxSize);

Console.WriteLine(" --- Describe all Auto Scaling groups to show the current state
of the group ---");
var groups = await AutoScaleMethods.DescribeAutoScalingGroupsAsync(client,
groupName);

DisplayGroupDetails(groups);

PressEnter();

Console.WriteLine(" --- Describe account limits ---");
await AutoScaleMethods.DescribeAccountLimitsAsync(client);

Console.WriteLine("Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
System.Threading.Thread.Sleep(60000);

Console.WriteLine(" --- Set desired capacity to 2 ---");
int desiredCapacity = 2;
await AutoScaleMethods.SetDesiredCapacityAsync(client, groupName, desiredCapacity);

Console.WriteLine(" --- Get the two instance Id values and state ---");

// Empty the group before getting the details again.
groups.Clear();
groups = await AutoScaleMethods.DescribeAutoScalingGroupsAsync(client, groupName);
if (groups is not null)
{
    foreach (AutoScalingGroup group in groups)
    {
        Console.WriteLine($"The group name is {group.AutoScalingGroupName}");
        Console.WriteLine($"The group ARN is {group.AutoScalingGroupARN}");
        var instances = group.Instances;
```

```
        foreach (Instance instance in instances)
    {
        Console.WriteLine($"The instance id is {instance.InstanceId}");
        Console.WriteLine($"The lifecycle state is {instance.LifecycleState}");
    }
}

Console.WriteLine("**** List the scaling activities that have occurred for the
group");
var activities = await AutoScaleMethods.DescribeAutoScalingActivitiesAsync(client,
groupName);
if (activities is not null)
{
    activities.ForEach(activity =>
    {
        Console.WriteLine($"The activity Id is {activity.ActivityId}");
        Console.WriteLine($"The activity details are {activity.Details}");
    });
}

// Display the Amazon CloudWatch metrics that have been collected.
var metrics = await CloudWatchMethods.GetCloudWatchMetricsAsync(groupName);
Console.WriteLine($"Metrics collected for {groupName}:");
metrics.ForEach(metric =>
{
    Console.Write($"Metric name: {metric.MetricName}\t");
    Console.WriteLine($"Namespace: {metric.Namespace}");
});

var dataPoints = await CloudWatchMethods.GetMetricStatisticsAsync(groupName);
Console.WriteLine("Details for the metrics collected:");
dataPoints.ForEach(detail =>
{
    Console.WriteLine(detail);
});

// Disable metrics collection.
Console.WriteLine("Disabling the collection of metrics for {groupName}.");
success = await AutoScaleMethods.DisableMetricsCollectionAsync(client, groupName);

if (success)
{
    Console.WriteLine($"Successfully stopped metrics collection for {groupName}.");
}
else
{
    Console.WriteLine($"Could not stop metrics collection for {groupName}.");
}

// Terminate all instances in the group.
Console.WriteLine(" --- Now terminating all instances in the AWS Auto Scaling group
---");

if (groups is not null)
{
    groups.ForEach(group =>
    {
        // Only delete instances in the AutoScaling group we created.
        if (group.AutoScalingGroupName == groupName)
        {
            group.Instances.ForEach(async instance =>
            {
                await
AutoScaleMethods.TerminateInstanceInAutoScalingGroupAsync(client,
instance.InstanceId);
```

```

        });
    });
}

// After all instances are terminated, delete the group.
Console.WriteLine("--- Deleting the Auto Scaling group ---");
await AutoScaleMethods.DeleteAutoScalingGroupAsync(client, groupName);

// Delete the launch template.
var deletedLaunchTemplateName = await
    EC2Methods.DeleteLaunchTemplateAsync(launchTemplateId);

if (deletedLaunchTemplateName == launchTemplateName)
{
    Console.WriteLine("Successfully deleted the launch template.");
}

Console.WriteLine("The demo is now concluded.");

void DisplayDescription()
{
    Console.WriteLine("This code example performs the following operations:");
    Console.WriteLine(" 1. Creates an Amazon EC2 launch template.");
    Console.WriteLine(" 2. Creates an Auto Scaling group.");
    Console.WriteLine(" 3. Shows the details of the new Auto Scaling group");
    Console.WriteLine("    to show that only one instance was created.");
    Console.WriteLine(" 4. Enables metrics collection.");
    Console.WriteLine(" 5. Updates the AWS Auto Scaling group to increase the");
    Console.WriteLine("    capacity to three.");
    Console.WriteLine(" 6. Describes Auto Scaling groups again to show the");
    Console.WriteLine("    current state of the group.");
    Console.WriteLine(" 7. Changes the desired capacity of the Auto Scaling");
    Console.WriteLine("    group to use an additional instance.");
    Console.WriteLine(" 8. Shows that there are now instances in the group.");
    Console.WriteLine(" 9. Lists the scaling activities that have occurred for the
group.");
    Console.WriteLine("10. Displays the Amazon CloudWatch metrics that have");
    Console.WriteLine("    been collected.");
    Console.WriteLine("11. Disables metrics collection.");
    Console.WriteLine("12. Terminates all instances in the Auto Scaling group.");
    Console.WriteLine("13. Deletes the Auto Scaling group.");
    Console.WriteLine("14. Deletes the Amazon EC2 launch template.");
}

void DisplayGroupDetails(List<AutoScalingGroup> groups)
{
    if (groups is null)
        return;

    groups.ForEach(group =>
    {
        Console.WriteLine($"Group name:\t{group.AutoScalingGroupName}");
        Console.WriteLine($"Group created:\t{group.CreatedTime}");
        Console.WriteLine($"Maximum number of instances:\t{group.MaxValue}");
        Console.WriteLine($"Desired number of instances:
\t{group.DesiredCapacity}");
    });
}

void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\n\n");
}

```

Define functions that are called by the scenario to manage launch templates and metrics. These functions wrap Amazon EC2 and CloudWatch actions.

```
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// The methods in this class create and delete an Amazon Elastic Compute
/// Cloud (Amazon EC2) launch template for use by the Amazon EC2 Auto
/// Scaling scenario.
/// </summary>
public class EC2Methods
{
    /// <summary>
    /// Create a new Amazon EC2 launch template.
    /// </summary>
    /// <param name="imageId">The image Id to use for instances launched
    /// using the Amazon EC2 launch template.</param>
    /// <param name="instanceType">The type of EC2 instances to create.</param>
    /// <param name="launchTemplateName">The name of the launch template.</param>
    /// <returns>Returns the TemplateID of the new launch template.</returns>
    public static async Task<string> CreateLaunchTemplateAsync(
        string imageId,
        string instanceType,
        string launchTemplateName)
    {
        var client = new AmazonEC2Client();

        var request = new CreateLaunchTemplateRequest
        {
            LaunchTemplateData = new RequestLaunchTemplateData
            {
                ImageId = imageId,
                InstanceType = instanceType,
            },
            LaunchTemplateName = launchTemplateName,
        };

        var response = await client.CreateLaunchTemplateAsync(request);

        return response.LaunchTemplate.LaunchTemplateId;
    }

    /// <summary>
    /// Deletes an Amazon EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateId">The TemplateId of the launch template to
    /// delete.</param>
    /// <returns>The name of the EC2 launch template that was deleted.</returns>
    public static async Task<string> DeleteLaunchTemplateAsync(string
        launchTemplateId)
    {
        var client = new AmazonEC2Client();

        var request = new DeleteLaunchTemplateRequest
        {
            LaunchTemplateId = launchTemplateId,
        };
    }
}
```

```
        var response = await client.DeleteLaunchTemplateAsync(request);
        return response.LaunchTemplate.LaunchTemplateName;
    }

    ///<summary>
    ///<summary>Retrieves a information about an EC2 launch template.
    ///</summary>
    ///<param name="launchTemplateName">The name of the EC2 launch template.</param>
    ///<returns>A Boolean value that indicates the success or failure of
    ///the operation.</returns>
    public static async Task<bool> DescribeLaunchTemplateAsync(string
launchTemplateName)
    {
        var client = new AmazonEC2Client();

        var request = new DescribeLaunchTemplatesRequest
        {
            LaunchTemplateNames = new List<string> { launchTemplateName, },
        };

        var response = await client.DescribeLaunchTemplatesAsync(request);

        if (response.LaunchTemplates != null)
        {
            response.LaunchTemplates.ForEach(template =>
            {
                Console.WriteLine($"{template.LaunchTemplateName}\t");
                Console.WriteLine(template.LaunchTemplateId);
            });

            return true;
        }

        return false;
    }
}

using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

    ///<summary>
    ///<summary>The method of this class display the metrics collected for the Amazon
    ///EC2 Auto Scaling group created by the Amazon EC2 Auto Scaling scenario.
    ///</summary>
    public class CloudWatchMethods
    {
        ///<summary>
        ///<summary>Retrieves the metrics information collection for the Auto Scaling
group.
        ///</summary>
        ///<param name="groupName">The name of the Auto Scaling group.</param>
        ///<returns>A list of Metrics collected for the Auto Scaling group.</returns>
        public static async Task<List<Metric>> GetCloudWatchMetricsAsync(string
groupName)
        {
            var client = new AmazonCloudWatchClient();

            var filter = new DimensionFilter
            {
                Name = "AutoScalingGroupName",
                Value = $"{groupName}",
            };
        }
    }
}
```

```
        var request = new ListMetricsRequest
    {
        MetricName = "AutoScalingGroupName",
        Dimensions = new List<DimensionFilter> { filter },
        Namespace = "AWS/AutoScaling",
    };

    var response = await client.ListMetricsAsync(request);

    return response.Metrics;
}

/// <summary>
/// Retrieves the metric data collected for an Amazon EC2 Auto Scaling
group.
/// </summary>
/// <param name="groupName">The name of the Amazon EC2 Auto Scaling
group.</param>
/// <returns>A list of data points.</returns>
public static async Task<List<Datapoint>> GetMetricStatisticsAsync(string
groupName)
{
    var client = new AmazonCloudWatchClient();

    var metricDimensions = new List<Dimension>
    {
        new Dimension
        {
            Name = "AutoScalingGroupName",
            Value = $"{groupName}",
        },
    };

    // The start time will be yesterday.
    var startTime = DateTime.UtcNow.AddDays(-1);

    var request = new GetMetricStatisticsRequest
    {
        MetricName = "AutoScalingGroupName",
        Dimensions = metricDimensions,
        Namespace = "AWS/AutoScaling",
        Period = 60, // 60 seconds
        Statistics = new List<string>() { "Minimum" },
        StartTimeUtc = startTime,
        EndTimeUtc = DateTime.UtcNow,
    };

    var response = await client.GetMetricStatisticsAsync(request);

    return response.Datapoints;
}
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [CreateAutoScalingGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAutoScalingInstances](#)
  - [DescribeScalingActivities](#)
  - [DisableMetricsCollection](#)
  - [EnableMetricsCollection](#)

- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
 * Before running this SDK for Java (v2) code example, set up your development  
environment, including your credentials.  
*  
* For more information, see the following documentation:  
*  
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
*  
* In addition, create a launch template. For more information, see the following  
topic:  
*  
* https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-templates.html#create-launch-template  
*  
* This code example performs the following operations:  
* 1. Creates an Auto Scaling group using an AutoScalingWaiter.  
* 2. Gets a specific Auto Scaling group and returns an instance Id value.  
* 3. Describes Auto Scaling with the Id value.  
* 4. Enables metrics collection.  
* 5. Update an Auto Scaling group.  
* 6. Describes Account details.  
* 7. Describe account details"  
* 8. Updates an Auto Scaling group to use an additional instance.  
* 9. Gets the specific Auto Scaling group and gets the number of instances.  
* 10. List the scaling activities that have occurred for the group.  
* 11. Terminates an instance in the Auto Scaling group.  
* 12. Stops the metrics collection.  
* 13. Deletes the Auto Scaling group.  
*/  
  
public class AutoScalingScenario {  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    public static void main(String[] args) throws InterruptedException {  
        final String usage = "\n" +  
            "Usage:\n" +  
            "  <groupName> <launchTemplateName> <serviceLinkedRoleARN>  
<vpcZoneId>\n\n" +  
            "Where:\n" +  
            "  groupName - The name of the Auto Scaling group.\n" +  
            "  launchTemplateName - The name of the launch template. \n" +  
            "  serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the  
service-linked role that the Auto Scaling group uses.\n" +  
            "  vpcZoneId - A subnet Id for a virtual private cloud (VPC) where  
instances in the Auto Scaling group can be created.\n" ;  
  
        if (args.length != 4) {  
            System.out.println(usage);  
        }  
    }  
}
```

```
        System.exit(1);
    }

    String groupName = args[0];
    String launchTemplateName = args[1];
    String serviceLinkedRoleARN = args[2];
    String vpcZoneId = args[3];
    AutoScalingClient autoScalingClient = AutoScalingClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an Auto Scaling group named "+groupName);
    createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
serviceLinkedRoleARN, vpcZoneId);
    System.out.println("Wait 1 min for the resources, including the instance.
Otherwise, an empty instance Id is returned");
    Thread.sleep(60000);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. Get Auto Scale group Id value");
    String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
    if (instanceId.compareTo("") ==0) {
        System.out.println("Error - no instance Id value");
        System.exit(1);
    } else {
        System.out.println("The instance Id value is "+instanceId);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Describe Auto Scaling with the Id value
"+instanceId);
    describeAutoScalingInstance( autoScalingClient, instanceId);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Enable metrics collection "+instanceId);
    enableMetricsCollection(autoScalingClient, groupName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("5. Update an Auto Scaling group to update max size to
3");
    updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
serviceLinkedRoleARN);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Describe Auto Scaling groups");
    describeAutoScalingGroups(autoScalingClient, groupName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("7. Describe account details");
    describeAccountLimits(autoScalingClient);
    System.out.println("Wait 1 min for the resources, including the instance.
Otherwise, an empty instance Id is returned");
```

```
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred for
the group");
        describeScalingActivities(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group");
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed." );
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
            .autoScalingGroupName(groupName)
            .maxRecords(10)
            .build();

        DescribeScalingActivitiesResponse response =
autoScalingClient.describeScalingActivities(scalingActivitiesRequest);
        List<Activity> activities = response.activities();
        for (Activity activity: activities) {
            System.out.println("The activity Id is "+activity.activityId());
            System.out.println("The activity details are "+activity.details());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
```

```

        try {
            SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                .autoScalingGroupName(groupName)
                .desiredCapacity(2)
                .build();

            autoScalingClient.setDesiredCapacity(capacityRequest);
            System.out.println("You have set the DesiredCapacity to 2");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
                                              String groupName,
                                              String launchTemplateName,
                                              String serviceLinkedRoleARN,
                                              String vpcZoneId) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(launchTemplateName)
                .build();

            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .availabilityZones("us-east-1a")
                .launchTemplate(templateSpecification)
                .maxSize(1)
                .minSize(1)
                .vpcZoneIdentifier(vpcZoneId)
                .serviceLinkedRoleARN(serviceLinkedRoleARN)
                .build();

            autoScalingClient.createAutoScalingGroup(request);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter.waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingInstance( AutoScalingClient
autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest.builder()
                .instanceIds(id)
                .build();

            DescribeAutoScalingInstancesResponse response =
autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest);

```

```
        List<AutoScalingInstanceDetails> instances =
    response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance:instances ) {
            System.out.println("The instance lifecycle state is:
"+instance.lifecycleState());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .maxRecords(10)
        .build();

        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group: groups) {
            System.out.println("*** The service to use for the health checks:
"+ group.healthCheckType());
        }

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
    try{
        String instanceId = "";
        DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group: groups) {
            System.out.println("The group name is " +
group.autoScalingGroupName());
            System.out.println("The group ARN is " +
group.autoScalingGroupARN());
            List<Instance> instances = group.instances();

            for (Instance instance : instances) {
                instanceId = instance.instanceId();
                System.out.println("The instance id is " + instanceId);
                System.out.println("The lifecycle state is "
+instance.lifecycleState());
            }
        }

        return instanceId ;
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
    return "" ;
}

public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {

        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
    try {
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .build();

        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
        System.out.println("The disable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeAccountLimits(AutoScalingClient autoScalingClient) {
    try {
        DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
        System.out.println("The max number of auto scaling groups is
"+response.maxNumberOfAutoScalingGroups());
        System.out.println("The current number of auto scaling groups is
"+response.numberOfAutoScalingGroups());

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName, String launchTemplateName, String serviceLinkedRoleARN) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
```

```
LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(launchTemplateName)
    .build();

UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
    .maxSize(3)
    .serviceLinkedRoleARN(serviceLinkedRoleARN)
    .autoScalingGroupName(groupName)
    .launchTemplate(templateSpecification)
    .build();

autoScalingClient.updateAutoScalingGroup(groupRequest);
DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
    .autoScalingGroupNames(groupName)
    .build();

WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter.waitUntilGroupInService(groupsRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("You successfully updated the auto scaling group
"+groupName);

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId){
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
    .instanceId(instanceId)
    .shouldDecrementDesiredCapacity(false)
    .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance "+instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
    .autoScalingGroupName(groupName)
    .forceDelete(true)
    .build();

        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest) ;
        System.out.println("You successfully deleted "+groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [CreateAutoScalingGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAutoScalingInstances](#)
  - [DescribeScalingActivities](#)
  - [DisableMetricsCollection](#)
  - [EnableMetricsCollection](#)
  - [SetDesiredCapacity](#)
  - [TerminateInstanceInAutoScalingGroup](#)
  - [UpdateAutoScalingGroup](#)

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun main(args: Array<String>) {  
    val usage = """  
Usage:  
    <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>  
  
Where:  
    groupName - The name of the Auto Scaling group.  
    launchTemplateName - The name of the launch template.  
    serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-linked  
    role that the Auto Scaling group uses.  
    vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances  
    in the Auto Scaling group can be created.  
    """  
  
    if (args.size != 4) {  
        println(usage)  
        exitProcess(1)  
    }  
  
    val groupName = args[0]  
    val launchTemplateName = args[1]  
    val serviceLinkedRoleARN = args[2]  
    val vpcZoneId = args[3]  
  
    println("**** Create an Auto Scaling group named $groupName")  
    createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,  
    vpcZoneId)  
  
    println("Wait 1 min for the resources, including the instance. Otherwise, an  
    empty instance Id is returned")  
    delay(60000)
```

```
val instanceId = getSpecificAutoScaling(groupName)
if (instanceId.compareTo("") == 0) {
    println("Error - no instance Id value")
    exitProcess(1)
} else {
    println("The instance Id value is $instanceId")
}

println("**** Describe Auto Scaling with the Id value $instanceId")
describeAutoScalingInstance(instanceId)

println("**** Enable metrics collection $instanceId")
enableMetricsCollection(groupName)

println("**** Update an Auto Scaling group to maximum size of 3")
updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

println("**** Describe all Auto Scaling groups to show the current state of the
groups")
describeAutoScalingGroups(groupName)

println("**** Describe account details")
describeAccountLimits()

println("Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned")
delay(60000)

println("**** Set desired capacity to 2")
setDesiredCapacity(groupName)

println("**** Get the two instance Id values and state")
getAutoScalingGroups(groupName)

println("**** List the scaling activities that have occurred for the group")
describeScalingActivities(groupName)

println("**** Terminate an instance in the Auto Scaling group")
terminateInstanceInAutoScalingGroup(instanceId)

println("**** Stop the metrics collection")
disableMetricsCollection(groupName)

println("**** Delete the Auto Scaling group")
deleteSpecificAutoScalingGroup(groupName)
}

suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques = DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
        maxRecords = 10
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
${group.healthCheckType}")
        }
    }
}

suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest = DisableMetricsCollectionRequest {
        autoScalingGroupName = groupName
    }
}
```

```

        metrics = listOf("GroupMaxSize")
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest = DescribeScalingActivitiesRequest {
        autoScalingGroupName = groupName
        maxRecords = 10
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
        autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
        response.activities?.forEach { activity ->
            println("The activity Id is ${activity.activityId}")
            println("The activity details are ${activity.details}")
        }
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest = DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
        autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}

suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest = SetDesiredCapacityRequest {
        autoScalingGroupName = groupName
        desiredCapacity = 2
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}

suspend fun updateAutoScalingGroup(groupName: String, launchTemplateNameVal: String, serviceLinkedRoleARNVal: String) {
    val templateSpecification = LaunchTemplateSpecification {
        launchTemplateName = launchTemplateNameVal
    }

    val groupRequest = UpdateAutoScalingGroupRequest {
        maxSize = 3
        serviceLinkedRoleArn = serviceLinkedRoleARNVal
        autoScalingGroupName = groupName
    }
}

```

```

        launchTemplate = templateSpecification
    }

    val groupsRequestWaiter = DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group $groupName")
    }
}

suspend fun createAutoScalingGroup(groupName: String, launchTemplateNameVal: String, serviceLinkedRoleARNVal: String, vpcZoneIdVal: String) {
    val templateSpecification = LaunchTemplateSpecification {
        launchTemplateName = launchTemplateNameVal
    }

    val request = CreateAutoScalingGroupRequest {
        autoScalingGroupName = groupName
        availabilityZones = listOf("us-east-1a")
        launchTemplate = templateSpecification
        maxSize = 1
        minSize = 1
        vpcZoneIdentifier = vpcZoneIdVal
        serviceLinkedRoleArn = serviceLinkedRoleARNVal
    }

    // This object is required for the waiter call.
    val groupsRequestWaiter = DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}

suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest {
        instanceIds = listOf(id)
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
        autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest = EnableMetricsCollectionRequest {
        autoScalingGroupName = groupName
        metrics = listOf("GroupMaxSize")
        granularity = "1Minute"
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}

```

```

        }
    }

    suspend fun getSpecificAutoScaling(groupName: String): String {
        var instanceId = ""
        val scalingGroupsRequest = DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

        AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
            val response =
                autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
            response.autoScalingGroups?.forEach { group ->
                println("The group name is ${group.autoScalingGroupName}")
                println("The group ARN is ${group.autoScalingGroupArn}")

                group.instances?.forEach { instance ->
                    instanceId = instance.instanceId.toString()
                }
            }
        }
        return instanceId
    }

    suspend fun describeAccountLimits() {
        AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
            val response =
                autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
            println("The max number of Auto Scaling groups is
${response.maxNumberOfAutoScalingGroups}")
            println("The current number of Auto Scaling groups is
${response.numberOfAutoScalingGroups}")
        }
    }

    suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
        val request = TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

        AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
            autoScalingClient.terminateInstanceInAutoScalingGroup(request)
            println("You have terminated instance $instanceIdVal")
        }
    }

    suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
        val deleteAutoScalingGroupRequest = DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

        AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
            autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
            println("You successfully deleted $groupName")
        }
    }
}

```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - [CreateAutoScalingGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DescribeAutoScalingGroups](#)

- [DescribeAutoScalingInstances](#)
- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
def run_scenario(as_wrapper, svc_helper):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the Amazon EC2 Auto Scaling demo for managing groups and instances.")
    print('*'*88)

    print("This example requires a launch template that specifies how to create\n"
          "EC2 instances. You can use an existing template or create a new one.")
    template_name = q.ask(
        "Enter the name of an existing launch template or press Enter to create a new one: ")
    template = None
    if template_name:
        template = svc_helper.get_template(template_name)
    if template is None:
        inst_type = 't1.micro'
        ami_id = 'ami-0ca285d4c2cda3300'
        print("Let's create a launch template with the following specifications:")
        print(f"\tInstanceType: {inst_type}")
        print(f"\tAMI ID: {ami_id}")
        template_name = q.ask("Enter a name for the template: ", q.non_empty)
        template = svc_helper.create_template(template_name, inst_type, ami_id)
    print('*'*88)

    print("Let's create an Auto Scaling group.")
    group_name = q.ask("Enter a name for the group: ", q.non_empty)
    zones = svc_helper.get_availability_zones()
    print("EC2 instances can be created in the following Availability Zones:")
    for index, zone in enumerate(zones):
        print(f"\t{index+1}. {zone}")
    print(f"\t{len(zones)+1}. All zones")
    zone_sel = q.ask("Which zone do you want to use? ", q.is_int, q.in_range(1,
    len(zones)+1))
    group_zones = [zones[zone_sel-1]] if zone_sel <= len(zones) else zones
    print(f"Creating group {group_name}...")
    as_wrapper.create_group(group_name, group_zones, template_name, 1, 1)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    print("Created group:")
    pp(group)
    print("Waiting for instance to start...")
```

```
wait_for_instances(group_name, as_wrapper)
print('*'*88)

use_metrics = q.ask(
    "Do you want to collect metrics about Amazon EC2 Auto Scaling during this
demo (y/n)? ",
    q.is_yesno)
if use_metrics:
    as_wrapper.enable_metrics(
        group_name, [
            'GroupMinSize', 'GroupMaxSize', 'GroupDesiredCapacity',
            'GroupInServiceInstances', 'GroupTotalInstances'])
    print(f"Metrics enabled for {group_name}.")
print('*'*88)

print(f"Let's update the maximum number of instances in {group_name} from 1 to
3.")
q.ask("Press Enter when you're ready.")
as_wrapper.update_group(group_name, MaxSize=3)
group = as_wrapper.describe_group(group_name)
print("The group still has one running instance, but can have up to three:")
print_simplified_group(group)
print('*'*88)

print(f"Let's update the desired capacity of {group_name} from 1 to 2.")
q.ask("Press Enter when you're ready.")
as_wrapper.set_desired_capacity(group_name, 2)
wait(10)
group = as_wrapper.describe_group(group_name)
print("Here's the current state of the group:")
print_simplified_group(group)
print('*'*88)
print("Waiting for the new instance to start...")
instance_ids = wait_for_instances(group_name, as_wrapper)
print('*'*88)

print(f"Let's terminate one of the instances in {group_name}.")
print("Because the desired capacity is 2, another instance will start.")
print("The currently running instances are:")
for index, inst_id in enumerate(instance_ids):
    print(f"\t{index+1}. {inst_id}")
inst_sel = q.ask(
    "Which instance do you want to stop? ", q.is_int, q.in_range(1,
len(instance_ids)+1))
print(f"Stopping {instance_ids[inst_sel-1]}...")
as_wrapper.terminate_instance(instance_ids[inst_sel-1], False)
wait(10)
group = as_wrapper.describe_group(group_name)
print(f"Here's the state of {group_name}:")
print_simplified_group(group)
print("Waiting for the scaling activities to complete...")
wait_for_instances(group_name, as_wrapper)
print('*'*88)

print(f"Let's get a report of scaling activities for {group_name}.")
q.ask("Press Enter when you're ready.")
activities = as_wrapper.describe_scaling_activities(group_name)
print(f"Found {len(activities)} activities.\n"
      f"Activities are ordered with the most recent one first:")
for act in activities:
    pp(act)
print('*'*88)

if use_metrics:
    print("Let's look at CloudWatch metrics.")
    metric_namespace = 'AWS/AutoScaling'
```

```

metric_dimensions = [{'Name': 'AutoScalingGroupName', 'Value': group_name}]
print(f"The following metrics are enabled for {group_name}:")
done = False
while not done:
    metrics = svc_helper.get_metrics(metric_namespace, metric_dimensions)
    for index, metric in enumerate(metrics):
        print(f"\t{index+1}. {metric.name}")
    print(f"\t{len(metrics)+1}. None")
    metric_sel = q.ask(
        "Which metric do you want to see? ", q.is_int, q.in_range(1,
len(metrics)+1))
    if metric_sel < len(metrics)+1:
        span = 5
        metric = metrics[metric_sel - 1]
        print(f"Over the last {span} minutes, {metric.name} recorded:")
        # CloudWatch metric times are in the UTC+0 time zone.
        now = datetime.now(timezone.utc)
        metric_data = svc_helper.get_metric_statistics(
            metric_dimensions, metric, now-timedelta(minutes=span), now)
        pp(metric_data)
        if not q.ask("Do you want to see another metric (y/n)? ",
q.is_yesno):
            done = True
        else:
            done = True

    print(f"Let's clean up.")
    q.ask("Press Enter when you're ready.")
if use_metrics:
    print(f"Stopping metrics collection for {group_name}.")
    as_wrapper.disable_metrics(group_name)

    print("You must terminate all instances in the group before you can delete the
group.")
    print("Set minimum size to 0.")
    as_wrapper.update_group(group_name, MinSize=0)
    group = as_wrapper.describe_group(group_name)
    instance_ids = [inst['InstanceId'] for inst in group['Instances']]
    for inst_id in instance_ids:
        print(f"Stopping {inst_id}.")
        as_wrapper.terminate_instance(inst_id, True)
    print("Waiting for instances to stop...")
    wait_for_instances(group_name, as_wrapper)
    print(f"Deleting {group_name}.")
    as_wrapper.delete_group(group_name)
    print('*'*88)

    if template is not None:
        if q.ask(f"Do you want to delete launch template {template_name} used in
this demo (y/n)? "):
            svc_helper.delete_template(template_name)
            print("Template deleted.")

    print("\nThanks for watching!")
    print('*'*88)

if __name__ == '__main__':
    try:
        wrapper = AutoScalingWrapper(boto3.client('autoscaling'))
        helper = ServiceHelper(boto3.client('ec2'), boto3.resource('cloudwatch'))
        run_scenario(wrapper, helper)
    except Exception:
        logging.exception("Something went wrong with the demo!")

```

Define functions that are called by the scenario to manage launch templates and metrics. These functions wrap Amazon EC2 and CloudWatch actions.

```
class ServiceHelper:  
    """Encapsulates Amazon EC2 and CloudWatch actions for the example."""  
    def __init__(self, ec2_client, cloudwatch_resource):  
        """  
        :param ec2_client: A Boto3 Amazon EC2 client.  
        :param cloudwatch_resource: A Boto3 CloudWatch resource.  
        """  
        self.ec2_client = ec2_client  
        self.cloudwatch_resource = cloudwatch_resource  
  
    def get_template(self, template_name):  
        """  
        Gets a launch template. Launch templates specify configuration for  
        instances  
        that are launched by Amazon EC2 Auto Scaling.  
  
        :param template_name: The name of the template to look up.  
        :return: The template, if it exists.  
        """  
        try:  
            response =  
self.ec2_client.describe_launch_templates(LaunchTemplateNames=[template_name])  
            template = response['LaunchTemplates'][0]  
        except ClientError as err:  
            if err.response['Error']['Code'] ==  
'InvalidLaunchTemplateName.NotFoundException':  
                logger.warning("Launch template %s does not exist.", template_name)  
            else:  
                logger.error(  
                    "Couldn't verify launch template %s. Here's why: %s: %s",  
template_name,  
                    err.response['Error']['Code'], err.response['Error']  
['Message'])  
                raise  
        else:  
            return template  
  
    def create_template(self, template_name, inst_type, ami_id):  
        """  
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.  
  
        :param template_name: The name to give to the template.  
        :param inst_type: The type of the instance, such as t1.micro.  
        :param ami_id: The ID of the Amazon Machine Image (AMI) to use when  
creating  
            an instance.  
        :return: Information about the newly created template.  
        """  
        try:  
            response = self.ec2_client.create_launch_template(  
                LaunchTemplateName=template_name,  
                LaunchTemplateData={  
                    'InstanceType': inst_type,  
                    'ImageId': ami_id})  
            template = response['LaunchTemplate']  
        except ClientError as err:  
            logger.error(  
                "Couldn't create launch template %s. Here's why: %s: %s",  
template_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:
```

```
        return template

    def delete_template(self, template_name):
        """
        Deletes a launch template.

        :param template_name: The name of the template to delete.
        """
        try:

            self.ec2_client.delete_launch_template(LaunchTemplateName=template_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete launch template %s. Here's why: %s: %s",
                template_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise

    def get_availability_zones(self):
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
        client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone['ZoneName'] for zone in response['AvailabilityZones']]
        except ClientError as err:
            logger.error(
                "Couldn't get availability zones. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return zones

    def get_metrics(self, namespace, dimensions):
        """
        Gets a list of CloudWatch metrics filtered by namespace and dimensions.

        :param namespace: The namespace of the metrics to look up.
        :param dimensions: The dimensions of the metrics to look up.
        :return: The list of metrics.
        """
        try:
            metrics = list(self.cloudwatch_resource.metrics.filter(
                Namespace=namespace, Dimensions=dimensions))
        except ClientError as err:
            logger.error(
                "Couldn't get metrics for %s, %s. Here's why: %s: %s",
                namespace, dimensions,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return metrics

    @staticmethod
    def get_metric_statistics(dimensions, metric, start, end):
        """
        Gets statistics for a CloudWatch metric within a specified time span.

        :param dimensions: The dimensions of the metric.
        :param metric: The metric to look up.
        :param start: The start of the time span for retrieved metrics.
        :param end: The end of the time span for retrieved metrics.
        :return: The list of data points found for the specified metric.
        
```

```

"""
try:
    response = metric.get_statistics(
        Dimensions=dimensions, StartTime=start, EndTime=end, Period=60,
        Statistics=['Sum'])
    data = response['Datapoints']
except ClientError as err:
    logger.error(
        "Couldn't get statistics for metric %s. Here's why: %s: %s",
        metric.name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return data

def print_simplified_group(group):
    """
    Prints a subset of data for an Auto Scaling group.
    """
    print(group['AutoScalingGroupName'])
    print(f"\tLaunch template: {group['LaunchTemplate']['LaunchTemplateName']}")
    print(f"\tMin: {group['MinSize']}, Max: {group['MaxSize']}, Desired: "
          f"{group['DesiredCapacity']}")
    if group['Instances']:
        print(f"\tInstances:")
        for inst in group['Instances']:
            print(f"\t\t{inst['InstanceId']}: {inst['LifecycleState']}")

def wait_for_instances(group_name, as_wrapper):
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.
    """
    ready = False
    instance_ids = []
    instances = []
    while not ready:
        group = as_wrapper.describe_group(group_name)
        instance_ids = [i['InstanceId'] for i in group['Instances']]
        instances = as_wrapper.describe_instances(instance_ids) if instance_ids
    else []:
        if all([x['LifecycleState'] in ['Terminated', 'InService'] for x in
               instances]):
            ready = True
        else:
            wait(10)
    if instances:
        print(f"Here are the details of the instance{'s' if len(instances) > 1 else ''}:")
        for instance in instances:
            pp(instance)
    return instance_ids

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateAutoScalingGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAutoScalingInstances](#)
  - [DescribeScalingActivities](#)

- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## Code examples for Amazon ECR using AWS SDKs

The following code examples show how to use Amazon Elastic Container Registry with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon ECR using AWS SDKs \(p. 739\)](#)
  - List the image IDs for an Amazon ECR repository using an AWS SDK (p. 739)
  - List your Amazon ECR repositories using an AWS SDK (p. 740)

## Actions for Amazon ECR using AWS SDKs

The following code examples show how to use Amazon Elastic Container Registry with AWS SDKs. Each example calls an individual service function.

### Examples

- [List the image IDs for an Amazon ECR repository using an AWS SDK \(p. 739\)](#)
- [List your Amazon ECR repositories using an AWS SDK \(p. 740\)](#)

## List the image IDs for an Amazon ECR repository using an AWS SDK

The following code example shows how to list the image IDs for a repository.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_images(  
    client: &aws_sdk_ecr::Client,  
    repository: &str,  
) -> Result<(), aws_sdk_ecr::Error> {  
    let rsp = client  
        .list_images()  
        .repository_name(repository)  
        .send()  
        .await?;
```

```
let images = rsp.image_ids().unwrap_or_default();

println!("found {} images", images.len());

for image in images {
    println!(
        "image: {}:{}",
        image.image_tag().unwrap(),
        image.image_digest().unwrap()
    );
}

Ok(())
}
```

- For API details, see [ListImages](#) in *AWS SDK for Rust API reference*.

## List your Amazon ECR repositories using an AWS SDK

The following code example shows how to list your repositories.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_repos(client: &aws_sdk_ecr::Client) -> Result<(), aws_sdk_ecr::Error>
{
    let rsp = client.describe_repositories().send().await?;

    let repos = rsp.repositories().unwrap_or_default();

    println!("Found {} repositories:", repos.len());

    for repo in repos {
        println!(" ARN: {}", repo.repository_arn().unwrap());
        println!(" Name: {}", repo.repository_name().unwrap());
    }

    Ok(())
}
```

- For API details, see [DescribeRepositories](#) in *AWS SDK for Rust API reference*.

## Code examples for Amazon ECS using AWS SDKs

The following code examples show how to use Amazon Elastic Container Service with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon ECS using AWS SDKs \(p. 741\)](#)
  - Create an Amazon ECS cluster using an AWS SDK (p. 741)
  - Delete an Amazon ECS cluster using an AWS SDK (p. 741)
  - List your Amazon ECS clusters using an AWS SDK (p. 742)

## Actions for Amazon ECS using AWS SDKs

The following code examples show how to use Amazon Elastic Container Service with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create an Amazon ECS cluster using an AWS SDK \(p. 741\)](#)
- [Delete an Amazon ECS cluster using an AWS SDK \(p. 741\)](#)
- [List your Amazon ECS clusters using an AWS SDK \(p. 742\)](#)

## Create an Amazon ECS cluster using an AWS SDK

The following code example shows how to create an Amazon ECS cluster.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_cluster(client: &aws_sdk_ecs::Client, name: &str) -> Result<(),  
aws_sdk_ecs::Error> {  
    let cluster = client.create_cluster().cluster_name(name).send().await?  
    println!("cluster created: {:?}", cluster);  
  
    Ok(())  
}
```

- For API details, see [CreateCluster](#) in *AWS SDK for Rust API reference*.

## Delete an Amazon ECS cluster using an AWS SDK

The following code example shows how to delete an Amazon ECS cluster.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn remove_cluster(
    client: &aws_sdk_ecs::Client,
    name: &str,
) -> Result<(), aws_sdk_ecs::Error> {
    let cluster_deleted = client.delete_cluster().cluster(name).send().await?;
    println!("cluster deleted: {:?}", cluster_deleted);

    Ok(())
}
```

- For API details, see [DeleteCluster](#) in *AWS SDK for Rust API reference*.

## List your Amazon ECS clusters using an AWS SDK

The following code example shows how to list your Amazon ECS clusters.

Rust

### SDK for Rust

**Note**

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_clusters(client: &aws_sdk_ecs::Client) -> Result<(), aws_sdk_ecs::Error> {
    let resp = client.describe_clusters().send().await?;

    let clusters = resp.clusters().unwrap_or_default();
    println!("Found {} clusters:", clusters.len());

    for cluster in clusters {
        println!("  ARN: {}", cluster.cluster_arn().unwrap());
        println!("  Name: {}", cluster.cluster_name().unwrap());
    }
    Ok(())
}
```

- For API details, see [DescribeClusters](#) in *AWS SDK for Rust API reference*.

## Code examples for Amazon EKS using AWS SDKs

The following code examples show how to use Amazon Elastic Kubernetes Service with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon EKS using AWS SDKs \(p. 743\)](#)

- Create an Amazon EKS cluster control plane using an AWS SDK (p. 743)
- Delete an Amazon EKS cluster control plane using an AWS SDK (p. 743)

## Actions for Amazon EKS using AWS SDKs

The following code examples show how to use Amazon Elastic Kubernetes Service with AWS SDKs. Each example calls an individual service function.

### Examples

- Create an Amazon EKS cluster control plane using an AWS SDK (p. 743)
- Delete an Amazon EKS cluster control plane using an AWS SDK (p. 743)

## Create an Amazon EKS cluster control plane using an AWS SDK

The following code example shows how to create an Amazon EKS cluster control plane.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_cluster(
    client: &aws_sdk_eks::Client,
    name: &str,
    arn: &str,
    subnet_ids: Vec<String>,
) -> Result<(), aws_sdk_eks::Error> {
    let cluster = client
        .create_cluster()
        .name(name)
        .role_arn(arn)
        .resources_vpc_config(
            VpcConfigRequest::builder()
                .set_subnet_ids(Some(subnet_ids))
                .build(),
        )
        .send()
        .await?;
    println!("cluster created: {:?}", cluster);
    Ok(())
}
```

- For API details, see [CreateCluster](#) in *AWS SDK for Rust API reference*.

## Delete an Amazon EKS cluster control plane using an AWS SDK

The following code example shows how to delete an Amazon EKS cluster.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn remove_cluster(
    client: &aws_sdk_eks::Client,
    name: &str,
) -> Result<(), aws_sdk_eks::Error> {
    let cluster_deleted = client.delete_cluster().name(name).send().await?;
    println!("cluster deleted: {:?}", cluster_deleted);

    Ok(())
}
```

- For API details, see [DeleteCluster](#) in *AWS SDK for Rust API reference*.

## Code examples for Amazon EMR using AWS SDKs

The following code examples show how to use Amazon EMR with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon EMR using AWS SDKs \(p. 744\)](#)
  - [Add steps to an Amazon EMR job flow using an AWS SDK \(p. 745\)](#)
  - [Describe an Amazon EMR cluster using an AWS SDK \(p. 746\)](#)
  - [Describe a step on an Amazon EMR cluster using an AWS SDK \(p. 747\)](#)
  - [List steps for an Amazon EMR cluster using an AWS SDK \(p. 748\)](#)
  - [Run an Amazon EMR job flow using an AWS SDK \(p. 748\)](#)
  - [Terminate Amazon EMR job flows using an AWS SDK \(p. 750\)](#)
- [Scenarios for Amazon EMR using AWS SDKs \(p. 750\)](#)
  - [Run a shell script to install libraries on Amazon EMR instances using an AWS SDK \(p. 750\)](#)
- [Cross-service examples for Amazon EMR using AWS SDKs \(p. 752\)](#)
  - [Create a long-lived Amazon EMR cluster and run several steps using an AWS SDK \(p. 752\)](#)
  - [Create a short-lived Amazon EMR cluster and run a step using an AWS SDK \(p. 752\)](#)

## Actions for Amazon EMR using AWS SDKs

The following code examples show how to use Amazon EMR with AWS SDKs. Each example calls an individual service function.

### Examples

- [Add steps to an Amazon EMR job flow using an AWS SDK \(p. 745\)](#)
- [Describe an Amazon EMR cluster using an AWS SDK \(p. 746\)](#)

- [Describe a step on an Amazon EMR cluster using an AWS SDK \(p. 747\)](#)
- [List steps for an Amazon EMR cluster using an AWS SDK \(p. 748\)](#)
- [Run an Amazon EMR job flow using an AWS SDK \(p. 748\)](#)
- [Terminate Amazon EMR job flows using an AWS SDK \(p. 750\)](#)

## Add steps to an Amazon EMR job flow using an AWS SDK

The following code example shows how to add steps to an Amazon EMR job flow.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Add a Spark step, which is run by the cluster as soon as it is added.

```
def add_step(cluster_id, name, script_uri, script_args, emr_client):  
    """  
        Adds a job step to the specified cluster. This example adds a Spark  
        step, which is run by the cluster as soon as it is added.  
  
        :param cluster_id: The ID of the cluster.  
        :param name: The name of the step.  
        :param script_uri: The URI where the Python script is stored.  
        :param script_args: Arguments to pass to the Python script.  
        :param emr_client: The Boto3 EMR client object.  
        :return: The ID of the newly added step.  
    """  
    try:  
        response = emr_client.add_job_flow_steps(  
            JobFlowId=cluster_id,  
            Steps=[{  
                'Name': name,  
                'ActionOnFailure': 'CONTINUE',  
                'HadoopJarStep': {  
                    'Jar': 'command-runner.jar',  
                    'Args': ['spark-submit', '--deploy-mode', 'cluster',  
                            script_uri, *script_args]  
                }  
            }])  
        step_id = response['StepIds'][0]  
        logger.info("Started step with ID %s", step_id)  
    except ClientError:  
        logger.exception("Couldn't start step %s with URI %s.", name, script_uri)  
        raise  
    else:  
        return step_id
```

Run an Amazon EMR File System (EMRFS) command as a job step on a cluster. This can be used to automate EMRFS commands on a cluster instead of running commands manually through an SSH connection.

```
import boto3  
from botocore.exceptions import ClientError  
  
def add_emrfs_step(command, bucket_url, cluster_id, emr_client):
```

```
"""
Add an EMRFS command as a job flow step to an existing cluster.

:param command: The EMRFS command to run.
:param bucket_url: The URL of a bucket that contains tracking metadata.
:param cluster_id: The ID of the cluster to update.
:param emr_client: The Boto3 Amazon EMR client object.
:return: The ID of the added job flow step. Status can be tracked by calling
        the emr_client.describe_step() function.
"""

job_flow_step = {
    'Name': 'Example EMRFS Command Step',
    'ActionOnFailure': 'CONTINUE',
    'HadoopJarStep': {
        'Jar': 'command-runner.jar',
        'Args': [
            '/usr/bin/emrfs',
            command,
            bucket_url
        ]
    }
}

try:
    response = emr_client.add_job_flow_steps(
        JobFlowId=cluster_id, Steps=[job_flow_step])
    step_id = response['StepIds'][0]
    print(f"Added step {step_id} to cluster {cluster_id}.")
except ClientError:
    print(f"Couldn't add a step to cluster {cluster_id}.")
    raise
else:
    return step_id

def usage_demo():
    emr_client = boto3.client('emr')
    # Assumes the first waiting cluster has EMRFS enabled and has created metadata
    # with the default name of 'EmrFSMetadata'.
    cluster = emr_client.list_clusters(ClusterStates=['WAITING'])['Clusters'][0]
    add_emrfs_step(
        'sync', 's3://elasticmapreduce/samples/cloudfront', cluster['Id'],
        emr_client)

if __name__ == '__main__':
    usage_demo()
```

- For API details, see [AddJobFlowSteps](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe an Amazon EMR cluster using an AWS SDK

The following code example shows how to describe an Amazon EMR cluster.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def describe_cluster(cluster_id, emr_client):
    """
    Gets detailed information about a cluster.

    :param cluster_id: The ID of the cluster to describe.
    :param emr_client: The Boto3 EMR client object.
    :return: The retrieved cluster information.
    """
    try:
        response = emr_client.describe_cluster(ClusterId=cluster_id)
        cluster = response['Cluster']
        logger.info("Got data for cluster %s.", cluster['Name'])
    except ClientError:
        logger.exception("Couldn't get data for cluster %s.", cluster_id)
        raise
    else:
        return cluster
```

- For API details, see [DescribeCluster](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a step on an Amazon EMR cluster using an AWS SDK

The following code example shows how to describe a step on an Amazon EMR cluster.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def describe_step(cluster_id, step_id, emr_client):
    """
    Gets detailed information about the specified step, including the current state
    of the step.

    :param cluster_id: The ID of the cluster.
    :param step_id: The ID of the step.
    :param emr_client: The Boto3 EMR client object.
    :return: The retrieved information about the specified step.
    """
    try:
        response = emr_client.describe_step(ClusterId=cluster_id, StepId=step_id)
        step = response['Step']
        logger.info("Got data for step %s.", step_id)
    except ClientError:
        logger.exception("Couldn't get data for step %s.", step_id)
        raise
    else:
        return step
```

- For API details, see [DescribeStep](#) in *AWS SDK for Python (Boto3) API Reference*.

## List steps for an Amazon EMR cluster using an AWS SDK

The following code example shows how to list steps for an Amazon EMR cluster.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_steps(cluster_id, emr_client):
    """
    Gets a list of steps for the specified cluster. In this example, all steps are
    returned, including completed and failed steps.

    :param cluster_id: The ID of the cluster.
    :param emr_client: The Boto3 EMR client object.
    :return: The list of steps for the specified cluster.
    """
    try:
        response = emr_client.list_steps(ClusterId=cluster_id)
        steps = response['Steps']
        logger.info("Got %s steps for cluster %s.", len(steps), cluster_id)
    except ClientError:
        logger.exception("Couldn't get steps for cluster %s.", cluster_id)
        raise
    else:
        return steps
```

- For API details, see [ListSteps](#) in *AWS SDK for Python (Boto3) API Reference*.

## Run an Amazon EMR job flow using an AWS SDK

The following code example shows how to run an Amazon EMR job flow.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def run_job_flow(
    name, log_uri, keep_alive, applications, job_flow_role, service_role,
    security_groups, steps, emr_client):
    """
    Runs a job flow with the specified steps. A job flow creates a cluster of
    instances and adds steps to be run on the cluster. Steps added to the cluster
    are run as soon as the cluster is ready.

    This example uses the 'emr-5.30.1' release. A list of recent releases can be
    found here:
        https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-release-
    components.html.
    """
    # ... (code for creating cluster and adding steps)
```

```

:param name: The name of the cluster.
:param log_uri: The URI where logs are stored. This can be an Amazon S3 bucket
URL,
        such as 's3://my-log-bucket'.
:param keep_alive: When True, the cluster is put into a Waiting state after all
steps are run. When False, the cluster terminates itself
when
        the step queue is empty.
:param applications: The applications to install on each instance in the
cluster,
        such as Hive or Spark.
:param job_flow_role: The IAM role assumed by the cluster.
:param service_role: The IAM role assumed by the service.
:param security_groups: The security groups to assign to the cluster instances.
        Amazon EMR adds all needed rules to these groups, so
        they can be empty if you require only the default
rules.
:param steps: The job flow steps to add to the cluster. These are run in order
        when the cluster is ready.
:param emr_client: The Boto3 EMR client object.
:return: The ID of the newly created cluster.
"""
try:
    response = emr_client.run_job_flow(
        Name=name,
        LogUri=log_uri,
        ReleaseLabel='emr-5.30.1',
        Instances={
            'MasterInstanceType': 'm5.xlarge',
            'SlaveInstanceType': 'm5.xlarge',
            'InstanceCount': 3,
            'KeepJobFlowAliveWhenNoSteps': keep_alive,
            'EmrManagedMasterSecurityGroup': security_groups['manager'].id,
            'EmrManagedSlaveSecurityGroup': security_groups['worker'].id,
        },
        Steps=[{
            'Name': step['name'],
            'ActionOnFailure': 'CONTINUE',
            'HadoopJarStep': {
                'Jar': 'command-runner.jar',
                'Args': ['spark-submit', '--deploy-mode', 'cluster',
                        step['script_uri'], *step['script_args']]
            }
        } for step in steps],
        Applications=[{
            'Name': app
        } for app in applications],
        JobFlowRole=job_flow_role.name,
        ServiceRole=service_role.name,
        EbsRootVolumeSize=10,
        VisibleToAllUsers=True
    )
    cluster_id = response['JobFlowId']
    logger.info("Created cluster %s.", cluster_id)
except ClientError:
    logger.exception("Couldn't create cluster.")
    raise
else:
    return cluster_id

```

- For API details, see [RunJobFlow in AWS SDK for Python \(Boto3\) API Reference](#).

## Terminate Amazon EMR job flows using an AWS SDK

The following code example shows how to terminate Amazon EMR job flows.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def terminate_cluster(cluster_id, emr_client):
    """
    Terminates a cluster. This terminates all instances in the cluster and cannot
    be undone. Any data not saved elsewhere, such as in an Amazon S3 bucket, is
    lost.

    :param cluster_id: The ID of the cluster to terminate.
    :param emr_client: The Boto3 EMR client object.
    """
    try:
        emr_client.terminate_job_flows(JobFlowIds=[cluster_id])
        logger.info("Terminated cluster %s.", cluster_id)
    except ClientError:
        logger.exception("Couldn't terminate cluster %s.", cluster_id)
        raise
```

- For API details, see [TerminateJobFlows](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios for Amazon EMR using AWS SDKs

The following code examples show how to use Amazon EMR with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Run a shell script to install libraries on Amazon EMR instances using an AWS SDK \(p. 750\)](#)

## Run a shell script to install libraries on Amazon EMR instances using an AWS SDK

The following code example shows how to use AWS Systems Manager to run a shell script on Amazon EMR instances to install additional libraries. This can be used to automate instance management instead of running commands manually through an SSH connection.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import argparse
import time
```

```
import boto3

def install_libraries_on_core_nodes(
    cluster_id, script_path, emr_client, ssm_client):
    """
    Copies and runs a shell script on the core nodes in the cluster.

    :param cluster_id: The ID of the cluster.
    :param script_path: The path to the script, typically an Amazon S3 object URL.
    :param emr_client: The Boto3 Amazon EMR client.
    :param ssm_client: The Boto3 AWS Systems Manager client.
    """
    core_nodes = emr_client.list_instances(
        ClusterId=cluster_id, InstanceGroupTypes=['CORE'])['Instances']
    core_instance_ids = [node['Ec2InstanceId'] for node in core_nodes]
    print(f"Found core instances: {core_instance_ids}.")

    commands = [
        # Copy the shell script from Amazon S3 to each node instance.
        f"aws s3 cp {script_path} /home/hadoop",
        # Run the shell script to install libraries on each node instance.
        "bash /home/hadoop/install_libraries.sh"]
    for command in commands:
        print(f"Sending '{command}' to core instances...")
        command_id = ssm_client.send_command(
            InstanceIds=core_instance_ids,
            DocumentName='AWS-RunShellScript',
            Parameters={"commands": [command]},
            TimeoutSeconds=3600)['Command']['CommandId']
        while True:
            # Verify the previous step succeeded before running the next step.
            cmd_result = ssm_client.list_commands(
                CommandId=command_id)['Commands'][0]
            if cmd_result['StatusDetails'] == 'Success':
                print(f"Command succeeded.")
                break
            elif cmd_result['StatusDetails'] in ['Pending', 'InProgress']:
                print(f"Command status is {cmd_result['StatusDetails']},
waiting...")
                time.sleep(10)
            else:
                print(f"Command status is {cmd_result['StatusDetails']},
quitting.")
                raise RuntimeError(
                    f"Command {command} failed to run.
                    Details: {cmd_result['StatusDetails']}")

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('cluster_id', help="The ID of the cluster.")
    parser.add_argument('script_path', help="The path to the script in Amazon S3.")
    args = parser.parse_args()

    emr_client = boto3.client('emr')
    ssm_client = boto3.client('ssm')

    install_libraries_on_core_nodes(
        args.cluster_id, args.script_path, emr_client, ssm_client)

if __name__ == '__main__':
    main()
```

- For API details, see [ListInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Cross-service examples for Amazon EMR using AWS SDKs

The following code examples show how to use Amazon EMR with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Create a long-lived Amazon EMR cluster and run several steps using an AWS SDK \(p. 752\)](#)
- [Create a short-lived Amazon EMR cluster and run a step using an AWS SDK \(p. 752\)](#)

## Create a long-lived Amazon EMR cluster and run several steps using an AWS SDK

The following code example shows how to create a long-lived Amazon EMR cluster and run several steps.

### Python

#### SDK for Python (Boto3)

Create a long-lived Amazon EMR cluster that uses Apache Spark to query historical Amazon review data from the [Amazon Customer Reviews Dataset](#). Run a job that gets data for top-rated products in specific categories that contain keywords in their product titles. Job results are written to an Amazon Simple Storage Service (Amazon S3) bucket.

- Create an Amazon S3 bucket and upload a job script.
- Create AWS Identity and Access Management (IAM) roles.
- Create Amazon Elastic Compute Cloud (Amazon EC2) security groups.
- Create a long-lived cluster and run several job steps.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon EC2
- Amazon EMR
- IAM
- Amazon S3

## Create a short-lived Amazon EMR cluster and run a step using an AWS SDK

The following code example shows how to create a short-lived Amazon EMR cluster that runs a step and automatically terminates after the step completes.

### Python

#### SDK for Python (Boto3)

Create a short-lived Amazon EMR cluster that estimates the value of pi using Apache Spark to parallelize a large number of calculations. The job writes output to Amazon EMR logs and

to an Amazon Simple Storage Service (Amazon S3) bucket. The cluster terminates itself after completing the job.

- Create an Amazon S3 bucket and upload a job script.
- Create AWS Identity and Access Management (IAM) roles.
- Create Amazon Elastic Compute Cloud (Amazon EC2) security groups.
- Create a short-lived cluster and run a single job step.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Amazon EC2
- Amazon EMR
- IAM
- Amazon S3

## Code examples for EventBridge using AWS SDKs

The following code examples show how to use Amazon EventBridge with an AWS software development kit (SDK).

#### **Code examples**

- [Actions for EventBridge using AWS SDKs \(p. 753\)](#)
  - Add a Lambda function target using an AWS SDK (p. 753)
  - Create an EventBridge scheduled rule using an AWS SDK (p. 756)
  - Delete an EventBridge scheduled rule using an AWS SDK (p. 759)
  - Send EventBridge events using an AWS SDK (p. 760)
- [Scenarios for EventBridge using AWS SDKs \(p. 764\)](#)
  - Create and trigger a rule in Amazon EventBridge using an AWS SDK (p. 764)
- [Cross-service examples for EventBridge using AWS SDKs \(p. 778\)](#)
  - Use scheduled events to invoke a Lambda function (p. 778)

## Actions for EventBridge using AWS SDKs

The following code examples show how to use Amazon EventBridge with AWS SDKs. Each example calls an individual service function.

#### **Examples**

- [Add a Lambda function target using an AWS SDK \(p. 753\)](#)
- [Create an EventBridge scheduled rule using an AWS SDK \(p. 756\)](#)
- [Delete an EventBridge scheduled rule using an AWS SDK \(p. 759\)](#)
- [Send EventBridge events using an AWS SDK \(p. 760\)](#)

## Add a Lambda function target using an AWS SDK

The following code examples show how to add an AWS Lambda function target to an Amazon EventBridge event.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutTargetsRequest.h>
#include <aws/events/model/PutTargetsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Add the target.

```
Aws::CloudWatchEvents::EventBridgeClient cwe;

Aws::CloudWatchEvents::Model::Target target;
target.SetArn(lambda_arn);
target.SetId(target_id);

Aws::CloudWatchEvents::Model::PutTargetsRequest request;
request.SetRule(rule_name);
request.AddTargets(target);

auto putTargetsOutcome = cwe.PutTargets(request);
if (!putTargetsOutcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch events target for rule "
        << rule_name << ":" <<
        putTargetsOutcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout <<
        "Successfully created CloudWatch events target for rule "
        << rule_name << std::endl;
}
```

- For API details, see [PutTargets](#) in [AWS SDK for C++ API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { EventBridgeClient } from "@aws-sdk/client-eventbridge";
// Set the AWS Region.
```

```
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon EventBridge service client object.
export const ebClient = new EventBridgeClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js.
import { PutTargetsCommand } from "@aws-sdk/client-eventbridge";
import { ebClient } from "./libs/eventBridgeClient.js";

// Set the parameters.
export const params = {
  Rule: "DEMO_EVENT",
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN", //LAMBDA_FUNCTION_ARN
      Id: "myCloudWatchEventsTarget",
    },
  ],
};

export const run = async () => {
  try {
    const data = await ebClient.send(new PutTargetsCommand(params));
    console.log("Success, target added; requestID: ", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For API details, see [PutTargets](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({apiVersion: '2015-10-07'});

var params = {
  Rule: 'DEMO_EVENT',
  Targets: [
    {
      Arn: 'LAMBDA_FUNCTION_ARN',
      Id: 'myEventBridgeTarget',
    }
  ]
};
```

```
ebevents.putTargets(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For API details, see [PutTargets](#) in *AWS SDK for JavaScript API Reference*.

## Create an EventBridge scheduled rule using an AWS SDK

The following code examples show how to create an Amazon EventBridge scheduled rule.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutRuleRequest.h>
#include <aws/events/model/PutRuleResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Create the rule.

```
Aws::CloudWatchEvents::EventBridgeClient cwe;
Aws::CloudWatchEvents::Model::PutRuleRequest request;
request.SetName(rule_name);
request.SetRoleArn(role_arn);
request.SetScheduleExpression("rate(5 minutes)");
request.SetState(Aws::CloudWatchEvents::Model::RuleState::ENABLED);

auto outcome = cwe.PutRule(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch events rule " <<
        rule_name << ":" << outcome.GetError().GetMessage() <<
        std::endl;
}
else
{
    std::cout << "Successfully created CloudWatch events rule " <<
        rule_name << " with resulting Arn " <<
        outcome.GetResult().GetRuleArn() << std::endl;
}
```

- For API details, see [PutRule](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createEBRule(EventBridgeClient eventBrClient, String
ruleName, String cronExpression) {

    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .name(ruleName)
            .eventBusName("default")
            .scheduleExpression(cronExpression)
            .state("ENABLED")
            .description("A test rule that runs on a schedule created by
the Java API")
            .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is "+
ruleResponse.ruleArn());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutRule](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { EventBridgeClient } from "@aws-sdk/client-eventbridge";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon EventBridge service client object.
export const ebClient = new EventBridgeClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js.
import { PutRuleCommand } from "@aws-sdk/client-eventbridge";
import { ebClient } from "./libs/eventBridgeClient.js";
```

```
// Set the parameters.
export const params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN", //IAM_ROLE_ARN
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

export const run = async () => {
  try {
    const data = await ebClient.send(new PutRuleCommand(params));
    console.log("Success, scheduled rule created; Rule ARN:", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For API details, see [PutRule](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({apiVersion: '2015-10-07'});

var params = {
  Name: 'DEMO_EVENT',
  RoleArn: 'IAM_ROLE_ARN',
  ScheduleExpression: 'rate(5 minutes)',
  State: 'ENABLED'
};

ebevents.putRule(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.RuleArn);
  }
});
```

- For API details, see [PutRule](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createScRule(ruleName: String?, cronExpression: String?) {
    val ruleRequest = PutRuleRequest {
        name = ruleName
        eventBusName = "default"
        scheduleExpression = cronExpression
        state = RuleState.Enabled
        description = "A test rule that runs on a schedule created by the Kotlin
API"
    }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

- For API details, see [PutRule](#) in *AWS SDK for Kotlin API reference*.

## Delete an EventBridge scheduled rule using an AWS SDK

The following code examples show how to delete an Amazon EventBridge scheduled rule.

Java

### SDK for Java 2.x

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteEBRule(EventBridgeClient eventBrClient, String
ruleName) {

    try {
        DisableRuleRequest disableRuleRequest = DisableRuleRequest.builder()
            .name(ruleName)
            .eventBusName("default")
            .build();

        eventBrClient.disableRule(disableRuleRequest);
        DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
            .name(ruleName)
            .eventBusName("default")
            .build();

        eventBrClient.deleteRule(ruleRequest);
        System.out.println("Rule "+ruleName + " was successfully deleted!");

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteRule](#) in *AWS SDK for Java 2.x API Reference*.

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteEBRule(ruleName: String) {  
  
    val request = DisableRuleRequest {  
        name = ruleName  
        eventBusName = "default"  
    }  
  
    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->  
        eventBrClient.disableRule(request)  
        val ruleRequest = DeleteRuleRequest {  
            name = ruleName  
            eventBusName = "default"  
        }  
  
        eventBrClient.deleteRule(ruleRequest)  
        println("Rule $ruleName was successfully deleted!")  
    }  
}
```

- For API details, see [DeleteRule](#) in *AWS SDK for Kotlin API reference*.

## Send EventBridge events using an AWS SDK

The following code examples show how to send Amazon EventBridge events.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>  
#include <aws/events/EventBridgeClient.h>  
#include <aws/events/model/PutEventsRequest.h>  
#include <aws/events/model/PutEventsResult.h>  
#include <aws/core/utils/Outcome.h>  
#include <iostream>
```

Send the event.

```
Aws::CloudWatchEvents::EventBridgeClient cwe;

Aws::CloudWatchEvents::Model::PutEventsRequestEntry event_entry;
event_entry.SetDetail(MakeDetails(event_key, event_value));
event_entry.SetDetailType("sampleSubmitted");
event_entry.AddResources(resource_arn);
event_entry.SetSource("aws-sdk-cpp-cloudwatch-example");

Aws::CloudWatchEvents::Model::PutEventsRequest request;
request.AddEntries(event_entry);

auto outcome = cwe.PutEvents(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to post CloudWatch event: " <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully posted CloudWatch event" << std::endl;
}
```

- For API details, see [PutEvents](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void putEBEvents(EventBridgeClient eventBrClient, String
resourceArn, String resourceArn2 ) {

    try {
        // Populate a List with the resource ARN values.
        List<String> resources = new ArrayList<>();
        resources.add(resourceArn);
        resources.add(resourceArn2);

        PutEventsRequestEntry reqEntry = PutEventsRequestEntry.builder()
            .resources(resources)
            .source("com.mycompany.myapp")
            .detailType("myDetailType")
            .detail("{ \"key1\": \"value1\", \"key2\": \"value2\" }")
            .build();

        PutEventsRequest eventsRequest = PutEventsRequest.builder()
            .entries(reqEntry)
            .build();

        PutEventsResponse result = eventBrClient.putEvents(eventsRequest);
        for (PutEventsResultEntry resultEntry : result.entries()) {
            if (resultEntry.eventId() != null) {
                System.out.println("Event Id: " + resultEntry.eventId());
            } else {
                System.out.println("Injection failed with Error Code: " +
resultEntry.errorCode());
            }
        }
    }
}
```

```
        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see [PutEvents](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { EventBridgeClient } from "@aws-sdk/client-eventbridge";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon EventBridge service client object.
export const ebClient = new EventBridgeClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js.
import { PutEventsCommand } from "@aws-sdk/client-eventbridge";
import { ebClient } from "./libs/eventBridgeClient.js";

// Set the parameters.
export const params = {
    Entries: [
        {
            Detail: '{ "key1": "value1", "key2": "value2" }',
            DetailType: "appRequestSubmitted",
            Resources: [
                "RESOURCE_ARN", //RESOURCE_ARN
            ],
            Source: "com.company.app",
        },
    ],
};

export const run = async () => {
    try {
        const data = await ebClient.send(new PutEventsCommand(params));
        console.log("Success, event sent; requestID:", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
// Uncomment this line to run execution within this file.
// run();
```

- For API details, see [PutEvents](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({apiVersion: '2015-10-07'});

var params = {
  Entries: [
    {
      Detail: '{ \"key1\": \"value1\", \"key2\": \"value2\" }',
      DetailType: 'appRequestSubmitted',
      Resources: [
        'RESOURCE_ARN',
      ],
      Source: 'com.company.app'
    }
  ]
};

ebevents.putEvents(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
  }
});
```

- For API details, see [PutEvents](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun putEEvents(resourceArn: String, resourceArn2: String) {

    // Populate a List with the resource ARN values.
    val resources0b = mutableListOf<String>()
    resources0b.add(resourceArn)
    resources0b.add(resourceArn2)

    val reqEntry = PutEventsRequestEntry {
        resources = resources0b
        source = "com.mycompany.myapp"
```

```
        detailType = "myDetailType"
        detail = "{ \"key1\": \"value1\", \"key2\": \"value2\" }"
    }

    val request = PutEventsRequest {
        entries = listOf(reqEntry)
    }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val response = eventBrClient.putEvents(request)
        response.entries?.forEach { resultEntry ->

            if (resultEntry.eventId != null) {
                println("Event Id is ${resultEntry.eventId}")
            } else {
                println("Injection failed with Error Code
${resultEntry.errorCode}")
            }
        }
    }
}
```

- For API details, see [PutEvents](#) in *AWS SDK for Kotlin API reference*.

## Scenarios for EventBridge using AWS SDKs

The following code examples show how to use Amazon EventBridge with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Create and trigger a rule in Amazon EventBridge using an AWS SDK \(p. 764\)](#)

## Create and trigger a rule in Amazon EventBridge using an AWS SDK

The following code example shows how to create and trigger a rule in Amazon EventBridge.

Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Call the functions in the correct order.

```
require "aws-sdk-sns"
require "aws-sdk-iam"
require "aws-sdk-cloudwatchevents"
require "aws-sdk-ec2"
require "aws-sdk-cloudwatch"
require "aws-sdk-cloudwatchlogs"
require "securerandom"
```

Checks whether the specified Amazon Simple Notification Service (Amazon SNS) topic exists among those provided to this function.

```
# Checks whether the specified Amazon SNS
# topic exists among those provided to this function.
# This is a helper function that is called by the topic_exists? function.
#
# @param topics [Array] An array of Aws::SNS::Types::Topic objects.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
#     puts 'Topic found.'
#   end
#
def topic_found?(topics, topic_arn)
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  return false
end
```

Checks whether the specified topic exists among those available to the caller in Amazon SNS.

```
# Checks whether the specified topic exists among those available to the
# caller in Amazon SNS.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   exit 1 unless topic_exists?(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def topic_exists?(sns_client, topic_arn)
  puts "Searching for topic with ARN '#{topic_arn}'..."
  response = sns_client.list_topics
  if response.topics.count.positive?
    if topic_found?(response.topics, topic_arn)
      puts "Topic found."
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.topics.count.positive?
        if topic_found?(response.topics, topic_arn)
          puts "Topic found."
          return true
        end
      end
    end
  end
  puts "Topic not found."
  return false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  return false
end
```

Create a topic in Amazon SNS and then subscribe an email address to receive notifications to that topic.

```
# Creates a topic in Amazon SNS
# and then subscribes an email address to receive notifications to that topic.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_name [String] The name of the topic to create.
# @param email_address [String] The email address of the recipient to notify.
# @return [String] The ARN of the topic that was created.
# @example
#   puts create_topic(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
  puts "Topic created with ARN '#{topic_response.topic_arn}'."
  subscription_response = sns_client.subscribe(
    topic_arn: topic_response.topic_arn,
    protocol: "email",
    endpoint: email_address,
    return_subscription_arn: true
  )
  puts "Subscription created with ARN " \
    "'#{subscription_response.subscription_arn}'. Have the owner of the " \
    "email address '#{email_address}' check their inbox in a few minutes " \
    "and confirm the subscription to start receiving notification emails."
  return topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  return "Error"
end
```

Check whether the specified AWS Identity and Access Management (IAM) role exists among those provided to this function.

```
# Checks whether the specified AWS Identity and Access Management (IAM)
# role exists among those provided to this function.
# This is a helper function that is called by the role_exists? function.
#
# @param roles [Array] An array of Aws::IAM::Role objects.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-east-1')
#   response = iam_client.list_roles
#   if role_found?(
#     response.roles,
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
#     puts 'Role found.'
#   end
def role_found?(roles, role_arn)
  roles.each do |role|
    return true if role.arn == role_arn
  end
  return false
end
```

Check whether the specified role exists among those available to the caller in IAM.

```
# Checks whether the specified role exists among those available to the
# caller in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   exit 1 unless role_exists?(

#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
def role_exists?(iam_client, role_arn)
  puts "Searching for role with ARN '#{role_arn}'..."
  response = iam_client.list_roles
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts "Role found."
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.roles.count.positive?
        if role_found?(response.roles, role_arn)
          puts "Role found."
          return true
        end
      end
    end
  end
  puts "Role not found."
  return false
rescue StandardError => e
  puts "Role not found: #{e.message}"
  return false
end
```

Create a role in IAM.

```
# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon EventBridge to allow
# that rule to operate within the caller's account.
# This role is designed to be used specifically by this code example.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to create.
# @return [String] The ARN of the role that was created.
# @example
#   puts create_role(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
      'Version': "2012-10-17",
      'Statement': [
        {
          'Sid': "",
          'Effect': "Allow",
          'Principal': {
```

```

        'Service': "events.amazonaws.com"
    },
    'Action': "sts:AssumeRole"
]
].to_json,
path: "/",
role_name: role_name
)
puts "Role created with ARN '#{response.role.arn}'."
puts "Adding access policy to role..."
iam_client.put_role_policy(
    policy_document: [
        'Version': "2012-10-17",
        'Statement': [
            {
                'Sid': "CloudWatchEventsFullAccess",
                'Effect': "Allow",
                'Resource': "*",
                'Action': "events:)"
            },
            {
                'Sid': "IAMPassRoleForCloudWatchEvents",
                'Effect': "Allow",
                'Resource': "arn:aws:iam::*:role/AWS_Events_Invoke_Targets",
                'Action': "iam:PassRole"
            }
        ]
    ].to_json,
    policy_name: "CloudWatchEventsPolicy",
    role_name: role_name
)
puts "Access policy added to role."
return response.role.arn
rescue StandardError => e
    puts "Error creating role or adding policy to it: #{e.message}"
    puts "If the role was created, you must add the access policy " \
        "to the role yourself, or delete the role yourself and try again."
    return "Error"
end

```

Checks whether the specified EventBridge rule exists among those provided to this function.

```

# Checks whether the specified Amazon EventBridge rule exists among
# those provided to this function.
# This is a helper function that is called by the rule_exists? function.
#
# @param rules [Array] An array of Aws::CloudWatchEvents::Types::Rule objects.
# @param rule_arn [String] The name of the rule to find.
# @return [Boolean] true if the name of the rule was found; otherwise, false.
# @example
#   cloudwatchevents_client = Aws::CloudWatch::Client.new(region: 'us-east-1')
#   response = cloudwatchevents_client.list_rules
#   if rule_found?(response.rules, 'aws-doc-sdk-examples-ec2-state-change')
#     puts 'Rule found.'
#   end
def rule_found?(rules, rule_name)
    rules.each do |rule|
        return true if rule.name == rule_name
    end
    return false
end

```

Checks whether the specified rule exists among those available to the caller in EventBridge.

```
# Checks whether the specified rule exists among those available to the
# caller in Amazon EventBridge.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(

#     Aws::CloudWatch::Client.new(region: 'us-east-1')
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
  puts "Searching for rule with name '#{rule_name}'..."
  response = cloudwatchevents_client.list_rules
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts "Rule found."
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.rules.count.positive?
        if rule_found?(response.rules, rule_name)
          puts "Rule found."
          return true
        end
      end
    end
  end
  puts "Rule not found."
  return false
rescue StandardError => e
  puts "Rule not found: #{e.message}"
  return false
end
```

Create a rule in EventBridge.

```
# Creates a rule in Amazon EventBridge.
# This rule is triggered whenever an available instance in
# Amazon EC2 changes to the specified state.
# This rule is designed to be used specifically by this code example.
#
# Prerequisites:
#
# - A role in AWS Identity and Access Management (IAM) that is designed
#   to be used specifically by this code example.
# - A topic in Amazon SNS.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to create.
# @param rule_description [String] Some description for this rule.
# @param instance_state [String] The state that available instances in
#   Amazon EC2 must change to, to
#   trigger this rule.
# @param role_arn [String] The Amazon Resource Name (ARN) of the IAM role.
# @param target_id [String] Some identifying string for the rule's target.
# @param topic_arn [String] The ARN of the Amazon SNS topic.
# @return [Boolean] true if the rule was created; otherwise, false.
```

```
# @example
#   exit 1 unless rule_created?
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     'Triggers when any available EC2 instance starts.',
#     'running',
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change',
#     'sns-topic',
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def rule_created?(  
  cloudwatchevents_client,  
  rule_name,  
  rule_description,  
  instance_state,  
  role_arn,  
  target_id,  
  topic_arn
)  
  puts "Creating rule with name '#{rule_name}'..."  
  put_rule_response = cloudwatchevents_client.put_rule(  
    name: rule_name,  
    description: rule_description,  
    event_pattern: {  
      'source': [  
        "aws.ec2"  
      ],  
      'detail-type': [  
        "EC2 Instance State-change Notification"  
      ],  
      'detail': {  
        'state': [  
          instance_state  
        ]  
      }  
    }.to_json,  
    state: "ENABLED",  
    role_arn: role_arn
  )  
  puts "Rule created with ARN '#{put_rule_response.rule_arn}'."  
  
  put_targets_response = cloudwatchevents_client.put_targets(  
    rule: rule_name,  
    targets: [  
      {  
        id: target_id,  
        arn: topic_arn
      }
    ]
  )
  if put_targets_response.key?(:failed_entry_count) &&
    put_targets_response.failed_entry_count > 0
    puts "Error(s) adding target to rule:  
    put_targets_response.failed_entries.each do |failure|
      puts failure.error_message
    end
    return false
  else
    return true
  end
rescue StandardError => e
  puts "Error creating rule or adding target to rule: #{e.message}"
  puts "If the rule was created, you must add the target " \
    "to the rule yourself, or delete the rule yourself and try again."
  return false
end
```

Check to see whether the specified log group exists among those available to the caller in Amazon CloudWatch Logs.

```
# Checks to see whether the specified log group exists among those available
# to the caller in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to find.
# @return [Boolean] true if the log group name was found; otherwise, false.
# @example
#   exit 1 unless log_group_exists?
#   Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#   'aws-doc-sdk-examples-cloudwatch-log'
# )
def log_group_exists?(cloudwatchlogs_client, log_group_name)
  puts "Searching for log group with name '#{log_group_name}'..."
  response = cloudwatchlogs_client.describe_log_groups(
    log_group_name_prefix: log_group_name
  )
  if response.log_groups.count.positive?
    response.log_groups.each do |log_group|
      if log_group.log_group_name == log_group_name
        puts "Log group found."
        return true
      end
    end
  end
  puts "Log group not found."
  return false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  return false
end
```

Create a log group in CloudWatch Logs.

```
# Creates a log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to create.
# @return [Boolean] true if the log group name was created; otherwise, false.
# @example
#   exit 1 unless log_group_created?
#   Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#   'aws-doc-sdk-examples-cloudwatch-log'
# )
def log_group_created?(cloudwatchlogs_client, log_group_name)
  puts "Attempting to create log group with the name '#{log_group_name}'..."
  cloudwatchlogs_client.create_log_group(log_group_name: log_group_name)
  puts "Log group created."
  return true
rescue StandardError => e
  puts "Error creating log group: #{e.message}"
  return false
end
```

Write an event to a log stream in CloudWatch Logs.

```
# Writes an event to a log stream in Amazon CloudWatch Logs.  
#  
# Prerequisites:  
#  
# - A log group in Amazon CloudWatch Logs.  
# - A log stream within the log group.  
#  
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized  
#   Amazon CloudWatch Logs client.  
# @param log_group_name [String] The name of the log group.  
# @param log_stream_name [String] The name of the log stream within  
#   the log group.  
# @param message [String] The message to write to the log stream.  
# @param sequence_token [String] If available, the sequence token from the  
#   message that was written immediately before this message. This sequence  
#   token is returned by Amazon CloudWatch Logs whenever you programmatically  
#   write a message to the log stream.  
# @return [String] The sequence token that is returned by  
#   Amazon CloudWatch Logs after successfully writing the message to the  
#   log stream.  
# @example  
#   puts log_event(  
#     Aws::EC2::Client.new(region: 'us-east-1'),  
#     'aws-doc-sdk-examples-cloudwatch-log'  
#     '2020/11/19/53f985be-199f-408e-9a45-fc242df41fEX',  
#     'Instance `i-033c48ef067af3dEX` restarted.',  
#     '495426724868310740095796045676567882148068632824696073EX'  
#   )  
def log_event(  
  cloudwatchlogs_client,  
  log_group_name,  
  log_stream_name,  
  message,  
  sequence_token  
)  
  puts "Attempting to log '#{message}' to log stream '#{log_stream_name}'..."  
  event = {  
    log_group_name: log_group_name,  
    log_stream_name: log_stream_name,  
    log_events: [  
      {  
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,  
        message: message  
      }  
    ]  
  }  
  unless sequence_token.empty?  
    event[:sequence_token] = sequence_token  
  end  
  
  response = cloudwatchlogs_client.put_log_events(event)  
  puts "Message logged."  
  return response.next_sequence_token  
rescue StandardError => e  
  puts "Message not logged: #{e.message}"  
end
```

Restart an Amazon Elastic Compute Cloud (Amazon EC2) instance and adds information about the related activity to a log stream in CloudWatch Logs.

```
# Restarts an Amazon EC2 instance  
# and adds information about the related activity to a log stream
```

```
# in Amazon CloudWatch Logs.  
#  
# Prerequisites:  
#  
# - The Amazon EC2 instance to restart.  
# - The log group in Amazon CloudWatch Logs to add related activity  
#   information to.  
#  
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.  
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client]  
#   An initialized Amazon CloudWatch Logs client.  
# @param instance_id [String] The ID of the instance.  
# @param log_group_name [String] The name of the log group.  
# @return [Boolean] true if the instance was restarted and the information  
#   was written to the log stream; otherwise, false.  
# @example  
#   exit 1 unless instance_restarted?  
#     Aws::EC2::Client.new(region: 'us-east-1'),  
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),  
#     'i-033c48ef067af3dEX',  
#     'aws-doc-sdk-examples-cloudwatch-log'  
#   )  
def instance_restarted?  
  ec2_client,  
  cloudwatchlogs_client,  
  instance_id,  
  log_group_name  
)  
  log_stream_name = "#{Time.now.year}/#{Time.now.month}/#{Time.now.day}/" \  
    "#{SecureRandom.uuid}"  
  cloudwatchlogs_client.create_log_stream(  
    log_group_name: log_group_name,  
    log_stream_name: log_stream_name  
)  
  sequence_token = ""  
  
  puts "Attempting to stop the instance with the ID '#{instance_id}'. "  
    "This might take a few minutes..."  
  ec2_client.stop_instances(instance_ids: [instance_id])  
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])  
  puts "Instance stopped."  
  sequence_token = log_event(  
    cloudwatchlogs_client,  
    log_group_name,  
    log_stream_name,  
    "Instance '#{instance_id}' stopped.",  
    sequence_token  
)  
  
  puts "Attempting to restart the instance. This might take a few minutes..."  
  ec2_client.start_instances(instance_ids: [instance_id])  
  ec2_client.wait_until(:instance_running, instance_ids: [instance_id])  
  puts "Instance restarted."  
  sequence_token = log_event(  
    cloudwatchlogs_client,  
    log_group_name,  
    log_stream_name,  
    "Instance '#{instance_id}' restarted.",  
    sequence_token  
)  
  
  return true  
rescue StandardError => e  
  puts "Error creating log stream or stopping or restarting the instance: " \  
    "#{e.message}"  
  log_event(
```

```
cloudwatchlogs_client,
log_group_name,
log_stream_name,
"Error stopping or starting instance '#{instance_id}': #{e.message}",
sequence_token
)
return false
end
```

Display information about activity for a rule in EventBridge.

```
# Displays information about activity for a rule in Amazon EventBridge.
#
# Prerequisites:
#
# - A rule in Amazon EventBridge.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example
#   display_rule_activity(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     Time.now - 600, # Start checking from 10 minutes ago.
#     Time.now, # Check up until now.
#     60 # Check every minute during those 10 minutes.
#   )
def display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
  end_time,
  period
)
  puts "Attempting to display rule activity..."
  response = cloudwatch_client.get_metric_statistics(
    namespace: "AWS/Events",
    metric_name: "Invocations",
    dimensions: [
      {
        name: "RuleName",
        value: rule_name
      }
    ],
    start_time: start_time,
    end_time: end_time,
    period: period,
    statistics: ["Sum"],
    unit: "Count"
  )

  if response.key?(:datapoints) && response.datapoints.count.positive?
    puts "The event rule '#{rule_name}' was triggered:"
    response.datapoints.each do |datapoint|
      puts "  #{datapoint.sum} time(s) at #{datapoint.timestamp}"
    end
  else
    puts "The event rule '#{rule_name}' was not triggered during the \"\\"
```

```
        "specified time period."
    end
rescue StandardError => e
    puts "Error getting information about event rule activity: #{e.message}"
end
```

Display log information for all of the log streams in a CloudWatch Logs log group.

```
# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example
#   display_log_data(
#     Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def display_log_data(cloudwatchlogs_client, log_group_name)
    puts "Attempting to display log stream data for the log group " \
        "named '#{log_group_name}'..."
    describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
        log_group_name: log_group_name,
        order_by: "LastEventTime",
        descending: true
    )
    if describe_log_streams_response.key?(:log_streams) &&
        describe_log_streams_response.log_streams.count.positive?
        describe_log_streams_response.log_streams.each do |log_stream|
            get_log_events_response = cloudwatchlogs_client.get_log_events(
                log_group_name: log_group_name,
                log_stream_name: log_stream.log_stream_name
            )
            puts "\nLog messages for '#{log_stream.log_stream_name}':"
            puts "-" * (log_stream.log_stream_name.length + 20)
            if get_log_events_response.key?(:events) &&
                get_log_events_response.events.count.positive?
                    get_log_events_response.events.each do |event|
                        puts event.message
                    end
                else
                    puts "No log messages for this log stream."
                end
            end
        end
    rescue StandardError => e
        puts "Error getting information about the log streams or their messages: " \
            "#{e.message}"
    end
```

Display a reminder to the caller to manually clean up any associated AWS resources that they no longer need.

```
# Displays a reminder to the caller to manually clean up any associated
# AWS resources that they no longer need.
```

```
#  
# @param topic_name [String] The name of the Amazon SNS topic.  
# @param role_name [String] The name of the IAM role.  
# @param rule_name [String] The name of the Amazon EventBridge rule.  
# @param log_group_name [String] The name of the Amazon CloudWatch Logs log group.  
# @param instance_id [String] The ID of the Amazon EC2 instance.  
# @example  
#   manual_cleanup_notice(  
#     'aws-doc-sdk-examples-topic',  
#     'aws-doc-sdk-examples-cloudwatch-events-rule-role',  
#     'aws-doc-sdk-examples-ec2-state-change',  
#     'aws-doc-sdk-examples-cloudwatch-log',  
#     'i-033c48ef067af3dEX'  
#   )  
def manual_cleanup_notice(  
  topic_name, role_name, rule_name, log_group_name, instance_id  
)  
  puts "-" * 10  
  puts "Some of the following AWS resources might still exist in your account."  
  puts "If you no longer want to use this code example, then to clean up"  
  puts "your AWS account and avoid unexpected costs, you might want to"  
  puts "manually delete any of the following resources if they exist:"  
  puts "- The Amazon SNS topic named '#{topic_name}'."  
  puts "- The IAM role named '#{role_name}'."  
  puts "- The Amazon EventBridge rule named '#{rule_name}'."  
  puts "- The Amazon CloudWatch Logs log group named '#{log_group_name}'."  
  puts "- The Amazon EC2 instance with the ID '#{instance_id}'."  
end  
  
# Full example call:  
def run_me  
  # Properties for the Amazon SNS topic.  
  topic_name = "aws-doc-sdk-examples-topic"  
  email_address = "mary@example.com"  
  # Properties for the IAM role.  
  role_name = "aws-doc-sdk-examples-cloudwatch-events-rule-role"  
  # Properties for the Amazon EventBridge rule.  
  rule_name = "aws-doc-sdk-examples-ec2-state-change"  
  rule_description = "Triggers when any available EC2 instance starts."  
  instance_state = "running"  
  target_id = "sns-topic"  
  # Properties for the Amazon EC2 instance.  
  instance_id = "i-033c48ef067af3dEX"  
  # Properties for displaying the event rule's activity.  
  start_time = Time.now - 600 # Go back over the past 10 minutes  
                           # (10 minutes * 60 seconds = 600 seconds).  
  end_time = Time.now  
  period = 60 # Look back every 60 seconds over the past 10 minutes.  
  # Properties for the Amazon CloudWatch Logs log group.  
  log_group_name = "aws-doc-sdk-examples-cloudwatch-log"  
  # AWS service clients for this code example.  
  region = "us-east-1"  
  sts_client = Aws::STS::Client.new(region: region)  
  sns_client = Aws::SNS::Client.new(region: region)  
  iam_client = Aws::IAM::Client.new(region: region)  
  cloudwatchevents_client = Aws::CloudWatchEvents::Client.new(region: region)  
  ec2_client = Aws::EC2::Client.new(region: region)  
  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)  
  cloudwatchlogs_client = Aws::CloudWatchLogs::Client.new(region: region)  
  
  # Get the caller's account ID for use in forming  
  # Amazon Resource Names (ARNs) that this code relies on later.  
  account_id = sts_client.get_caller_identity.account  
  
  # If the Amazon SNS topic doesn't exist, create it.  
  topic_arn = "arn:aws:sns:#{{region}}:#{{account_id}}:#{{topic_name}}"
```

```
unless topic_exists?(sns_client, topic_arn)
  topic_arn = create_topic(sns_client, topic_name, email_address)
  if topic_arn == "Error"
    puts "Could not create the Amazon SNS topic correctly. Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
    exit 1
  end
end

# If the IAM role doesn't exist, create it.
role_arn = "arn:aws:iam::#{account_id}:role/#{role_name}"
unless role_exists?(iam_client, role_arn)
  role_arn = create_role(iam_client, role_name)
  if role_arn == "Error"
    puts "Could not create the IAM role correctly. Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end

# If the Amazon EventBridge rule doesn't exist, create it.
unless rule_exists?(cloudwatchevents_client, rule_name)
  unless rule_created?(
    cloudwatchevents_client,
    rule_name,
    rule_description,
    instance_state,
    role_arn,
    target_id,
    topic_arn
  )
    puts "Could not create the Amazon EventBridge rule correctly. " \
      "Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end

# If the Amazon CloudWatch Logs log group doesn't exist, create it.
unless log_group_exists?(cloudwatchlogs_client, log_group_name)
  unless log_group_created?(cloudwatchlogs_client, log_group_name)
    puts "Could not create the Amazon CloudWatch Logs log group " \
      "correctly. Program stopped."
    manual_cleanup_notice(
      topic_name, role_name, rule_name, log_group_name, instance_id
    )
  end
end

# Restart the Amazon EC2 instance, which triggers the rule.
unless instance_restarted?(
  ec2_client,
  cloudwatchlogs_client,
  instance_id,
  log_group_name
)
  puts "Could not restart the instance to trigger the rule. " \
    "Continuing anyway to show information about the rule and logs..."
end

# Display how many times the rule was triggered over the past 10 minutes.
display_rule_activity()
```

```
cloudwatch_client,
rule_name,
start_time,
end_time,
period
)

# Display related log data in Amazon CloudWatch Logs.
display_log_data(cloudwatchlogs_client, log_group_name)

# Reminder the caller to clean up any AWS resources that are used
# by this code example and are no longer needed.
manual_cleanup_notice(
    topic_name, role_name, rule_name, log_group_name, instance_id
)
end

run_me if $PROGRAM_NAME == __FILE__
```

## Cross-service examples for EventBridge using AWS SDKs

The following code examples show how to use Amazon EventBridge with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Use scheduled events to invoke a Lambda function \(p. 778\)](#)

## Use scheduled events to invoke a Lambda function

The following code examples show how to create an AWS Lambda function invoked by an Amazon EventBridge scheduled event.

Java

### SDK for Java 2.x

Shows how to create an Amazon EventBridge scheduled event that invokes an AWS Lambda function. Configure EventBridge to use a cron expression to schedule when the Lambda function is invoked. In this example, you create a Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create an app that sends a mobile text message to your employees that congratulates them at the one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

## JavaScript

### SDK for JavaScript V3

Shows how to create an Amazon EventBridge scheduled event that invokes an AWS Lambda function. Configure EventBridge to use a cron expression to schedule when the Lambda function is invoked. In this example, you create a Lambda function by using the Lambda JavaScript runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create an app that sends a mobile text message to your employees that congratulates them at the one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

#### Services used in this example

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

## Python

### SDK for Python (Boto3)

This example shows how to register an AWS Lambda function as the target of a scheduled Amazon EventBridge event. The Lambda handler writes a friendly message and the full event data to Amazon CloudWatch Logs for later retrieval.

- Deploys a Lambda function.
- Creates an EventBridge scheduled event and makes the Lambda function the target.
- Grants permission to let EventBridge invoke the Lambda function.
- Prints the latest data from CloudWatch Logs to show the result of the scheduled invocations.
- Cleans up all resources created during the demo.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- CloudWatch Logs
- EventBridge
- Lambda

# Code examples for AWS Glue using AWS SDKs

The following code examples show how to use AWS Glue with an AWS software development kit (SDK).

## Code examples

- [Actions for AWS Glue using AWS SDKs \(p. 780\)](#)

- [Create an AWS Glue crawler using an AWS SDK \(p. 780\)](#)
- [Create an AWS Glue job definition using an AWS SDK \(p. 785\)](#)
- [Delete an AWS Glue crawler using an AWS SDK \(p. 788\)](#)
- [Delete a database from the AWS Glue Data Catalog using an AWS SDK \(p. 790\)](#)
- [Delete an AWS Glue job definition using an AWS SDK \(p. 792\)](#)
- [Delete a table from an AWS Glue Data Catalog database using an AWS SDK \(p. 794\)](#)
- [Get an AWS Glue crawler using an AWS SDK \(p. 796\)](#)
- [Get a database from the AWS Glue Data Catalog using an AWS SDK \(p. 799\)](#)
- [Get an AWS Glue job run using an AWS SDK \(p. 802\)](#)
- [Get database list from the AWS Glue Data Catalog using an AWS SDK \(p. 804\)](#)
- [Get a job from the AWS Glue Data Catalog using an AWS SDK \(p. 804\)](#)
- [Get runs of an AWS Glue job using an AWS SDK \(p. 805\)](#)
- [Get tables from a database in the AWS Glue Data Catalog using an AWS SDK \(p. 807\)](#)
- [List AWS Glue job definitions using an AWS SDK \(p. 810\)](#)
- [Start an AWS Glue crawler using an AWS SDK \(p. 811\)](#)
- [Start an AWS Glue job run using an AWS SDK \(p. 814\)](#)
- [Scenarios for AWS Glue using AWS SDKs \(p. 817\)](#)
  - [Get started running AWS Glue crawlers and jobs using an AWS SDK \(p. 817\)](#)

## Actions for AWS Glue using AWS SDKs

The following code examples show how to use AWS Glue with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create an AWS Glue crawler using an AWS SDK \(p. 780\)](#)
- [Create an AWS Glue job definition using an AWS SDK \(p. 785\)](#)
- [Delete an AWS Glue crawler using an AWS SDK \(p. 788\)](#)
- [Delete a database from the AWS Glue Data Catalog using an AWS SDK \(p. 790\)](#)
- [Delete an AWS Glue job definition using an AWS SDK \(p. 792\)](#)
- [Delete a table from an AWS Glue Data Catalog database using an AWS SDK \(p. 794\)](#)
- [Get an AWS Glue crawler using an AWS SDK \(p. 796\)](#)
- [Get a database from the AWS Glue Data Catalog using an AWS SDK \(p. 799\)](#)
- [Get an AWS Glue job run using an AWS SDK \(p. 802\)](#)
- [Get database list from the AWS Glue Data Catalog using an AWS SDK \(p. 804\)](#)
- [Get a job from the AWS Glue Data Catalog using an AWS SDK \(p. 804\)](#)
- [Get runs of an AWS Glue job using an AWS SDK \(p. 805\)](#)
- [Get tables from a database in the AWS Glue Data Catalog using an AWS SDK \(p. 807\)](#)
- [List AWS Glue job definitions using an AWS SDK \(p. 810\)](#)
- [Start an AWS Glue crawler using an AWS SDK \(p. 811\)](#)
- [Start an AWS Glue job run using an AWS SDK \(p. 814\)](#)

## Create an AWS Glue crawler using an AWS SDK

The following code examples show how to create an AWS Glue crawler.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Creates an AWS Glue crawler.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="iam">The Amazon Resource Name (ARN) of the IAM role
/// that is used by the crawler.</param>
/// <param name="s3Path">The path to the Amazon S3 bucket where
/// data is stored.</param>
/// <param name="cron">The name of the CRON job that runs the crawler.</param>
/// <param name="dbName">The name of the database.</param>
/// <param name="crawlerName">The name of the AWS Glue crawler.</param>
/// <returns>A Boolean value indicating whether the AWS Glue crawler was
/// created successfully.</returns>
public static async Task<bool> CreateGlueCrawlerAsync(
    AmazonGlueClient glueClient,
    string iam,
    string s3Path,
    string cron,
    string dbName,
    string crawlerName)
{
    var s3Target = new S3Target
    {
        Path = s3Path,
    };

    var targetList = new List<S3Target>
    {
        s3Target,
    };

    var targets = new CrawlerTargets
    {
        S3Targets = targetList,
    };

    var crawlerRequest = new CreateCrawlerRequest
    {
        DatabaseName = dbName,
        Name = crawlerName,
        Description = "Created by the AWS Glue .NET API",
        Targets = targets,
        Role = iam,
        Schedule = cron,
    };

    var response = await glueClient.CreateCrawlerAsync(crawlerRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{crawlerName} was successfully created");
        return true;
    }
    else
    {
```

```
        Console.WriteLine($"Could not create {crawlerName}.");
        return false;
    }
}
```

- For API details, see [CreateCrawler](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createGlueCrawler(GlueClient glueClient,
                                      String iam,
                                      String s3Path,
                                      String cron,
                                      String dbName,
                                      String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        // Add the S3Target to a list.
        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);

        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
            .schedule(cron)
            .build();

        glueClient.createCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully created");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateCrawler](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: [
      S3Targets: [{ Path: s3TargetPath }],
    ],
  });

  return client.send(command);
};
```

- For API details, see [CreateCrawler](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createGlueCrawler(
  iam: String?,
  s3Path: String?,
  cron: String?,
  dbName: String?,
  crawlerName: String
) {
  val s3Target = S3Target {
    path = s3Path
  }

  // Add the S3Target to a list.
  val targetList = mutableListOf<S3Target>()
  targetList.add(s3Target)

  val target0b = CrawlerTargets {
    s3Targets = targetList
  }

  val request = CreateCrawlerRequest {
    databaseName = dbName
    name = crawlerName
  }
}
```

```
        description = "Created by the AWS Glue Kotlin API"
        targets = target0b
        role = iam
        schedule = cron
    }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

- For API details, see [CreateCrawler](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$crawlerName = "example-crawler-test-" . $uniqid;

$role = $iamService->getRole("AWSGlueServiceRole-DocExample");

$path = 's3://crawler-public-us-east-1/flight/2016/csv';
$glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);

public function createCrawler($crawlerName, $role, $databaseName, $path):
Result
{
    return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
        return $this->glueClient->createCrawler([
            'Name' => $crawlerName,
            'Role' => $role,
            'DatabaseName' => $databaseName,
            'Targets' => [
                'S3Targets' =>
                [
                    'Path' => $path,
                ]
            ],
        ]);
    });
}
```

- For API details, see [CreateCrawler](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):  
        """  
        Creates a crawler that can crawl the specified target and populate a  
        database in your AWS Glue Data Catalog with metadata that describes the  
        data  
        in the target.  
  
        :param name: The name of the crawler.  
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and  
        Access  
               Management (IAM) role that grants permission to let AWS  
        Glue  
               access the resources it needs.  
        :param db_name: The name to give the database that is created by the  
        crawler.  
        :param db_prefix: The prefix to give any database tables that are created  
        by  
               the crawler.  
        :param s3_target: The URL to an S3 bucket that contains data that is  
        the target of the crawler.  
        """  
        try:  
            self.glue_client.create_crawler(  
                Name=name,  
                Role=role_arn,  
                DatabaseName=db_name,  
                TablePrefix=db_prefix,  
                Targets={'S3Targets': [{'Path': s3_target}]}))  
        except ClientError as err:  
            logger.error(  
                "Couldn't create crawler. Here's why: %s: %s",  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise
```

- For API details, see [CreateCrawler](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an AWS Glue job definition using an AWS SDK

The following code examples show how to create an AWS Glue job definition.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Creates an AWS Glue job.  
/// </summary>
```

```

    ///<param name="glueClient">The initialized AWS Glue client.</param>
    ///<param name="jobName">The name of the job to create.</param>
    ///<param name="iam">The Amazon Resource Name (ARN) of the IAM role
    ///that will be used by the job.</param>
    ///<param name="scriptLocation">The location where the script is stored.</param>
    ///<returns>A Boolean value indicating whether the AWS Glue job was
    ///created successfully.</returns>
    public static async Task<bool> CreateJobAsync(AmazonGlueClient glueClient,
    string jobName, string iam, string scriptLocation)
    {
        var command = new JobCommand
        {
            PythonVersion = "3",
            Name = "MyJob1",
            ScriptLocation = scriptLocation,
        };

        var jobRequest = new CreateJobRequest
        {
            Description = "A Job created by using the AWS SDK for .NET",
            GlueVersion = "2.0",
            WorkerType = WorkerType.G1X,
            NumberOfWorkers = 10,
            Name = jobName,
            Role = iam,
            Command = command,
        };

        var response = await glueClient.CreateJobAsync(jobRequest);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"'{jobName}' was successfully created.");
            return true;
        }

        Console.WriteLine($"'{jobName}' could not be created.");
        return false;
    }

```

- For API details, see [CreateJob](#) in *AWS SDK for .NET API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    }
  });

```

```
        },
        GlueVersion: "3.0",
    });

    return client.send(command);
};
```

- For API details, see [CreateJob](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$role = $iamService->getRole("AWSGlueServiceRole-DocExample");

$jobName = 'test-job-' . $unqid;

$scriptLocation = "s3://$bucketName/run_job.py";
$job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
{
    return $this->glueClient->createJob([
        'Name' => $jobName,
        'Role' => $role,
        'Command' => [
            'Name' => 'glueetl',
            'ScriptLocation' => $scriptLocation,
            'PythonVersion' => $pythonVersion,
        ],
        'GlueVersion' => $glueVersion,
    ]);
}
```

- For API details, see [CreateJob](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client
```

```
def create_job(self, name, description, role_arn, script_location):
    """
    Creates a job definition for an extract, transform, and load (ETL) job that
    can be run by AWS Glue.

    :param name: The name of the job definition.
    :param description: The description of the job definition.
    :param role_arn: The ARN of an IAM role that grants AWS Glue the
        permissions
            it requires to run the job.
    :param script_location: The Amazon S3 URL of a Python ETL script that is
        run as
            part of the job. The script defines how the data is
        transformed.
    """
    try:
        self.glue_client.create_job(
            Name=name, Description=description, Role=role_arn,
            Command={'Name': 'glueetl', 'ScriptLocation': script_location,
            'PythonVersion': '3'},
            GlueVersion='3.0')
    except ClientError as err:
        logger.error(
            "Couldn't create job %s. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
```

- For API details, see [CreateJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an AWS Glue crawler using an AWS SDK

The following code examples show how to delete an AWS Glue crawler.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Deletes the named AWS Glue crawler.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="crawlerName">The name of the crawler to delete.</param>
/// <returns>A Boolean value indicating whether the AWS Glue crawler was
/// deleted successfully.</returns>
public static async Task<bool> DeleteSpecificCrawlerAsync(AmazonGlueClient
glueClient, string crawlerName)
{
    var deleteCrawlerRequest = new DeleteCrawlerRequest
    {
        Name = crawlerName,
    };

    var response = await
glueClient.DeleteCrawlerAsync(deleteCrawlerRequest);
```

```
        if (response.StatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"'{crawlerName}' was deleted");
        return true;
    }

    Console.WriteLine($"Could not create '{crawlerName}'.");
    return false;
}
```

- For API details, see [DeleteCrawler](#) in *AWS SDK for .NET API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const deleteCrawler = (crawlerName) => {
    const client = new GlueClient({ region: DEFAULT_REGION });

    const command = new DeleteCrawlerCommand({
        Name: crawlerName,
    });

    return client.send(command);
};
```

- For API details, see [DeleteCrawler](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
echo "Delete the crawler.\n";
$glueClient->deleteCrawler([
    'Name' => $crawlerName,
]);

public function deleteCrawler($crawlerName)
{
    return $this->glueClient->deleteCrawler([
        'Name' => $crawlerName,
    ]);
}
```

- For API details, see [DeleteCrawler](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def delete_crawler(self, name):  
        """  
        Deletes a crawler.  
  
        :param name: The name of the crawler to delete.  
        """  
        try:  
            self.glue_client.delete_crawler(Name=name)  
        except ClientError as err:  
            logger.error(  
                "Couldn't delete crawler %s. Here's why: %s: %s", name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise
```

- For API details, see [DeleteCrawler](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a database from the AWS Glue Data Catalog using an AWS SDK

The following code examples show how to delete a database from the AWS Glue Data Catalog.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Deletes an AWS Glue database.  
/// </summary>  
/// <param name="glueClient">The initialized AWS Glue client.</param>  
/// <param name="databaseName">The name of the database to delete.</param>  
/// <returns>A Boolean value indicating whether the AWS Glue database was  
/// deleted successfully.</returns>  
public static async Task<bool> DeleteDatabaseAsync(AmazonGlueClient  
glueClient, string databaseName)  
{  
    var request = new DeleteDatabaseRequest  
    {  
        Name = databaseName,
```

```
};

var response = await glueClient.DeleteDatabaseAsync(request);

if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
{
    Console.WriteLine($"{databaseName} was successfully deleted");
    return true;
}

Console.WriteLine($"{databaseName} could not be deleted.");
return false;
}
```

- For API details, see [DeleteDatabase](#) in *AWS SDK for .NET API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const deleteDatabase = (databaseName) => {
    const client = new GlueClient({ region: DEFAULT_REGION });

    const command = new DeleteDatabaseCommand({
        Name: databaseName,
    });

    return client.send(command);
};
```

- For API details, see [DeleteDatabase](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
echo "Delete the databases.\n";
$glueClient->deleteDatabase([
    'Name' => $databaseName,
]);

public function deleteDatabase($databaseName)
{
    return $this->glueClient->deleteDatabase([
        'Name' => $databaseName,
    ]);
}
```

- For API details, see [DeleteDatabase](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def delete_database(self, name):  
        """  
        Deletes a metadata database from your Data Catalog.  
  
        :param name: The name of the database to delete.  
        """  
        try:  
            self.glue_client.delete_database(Name=name)  
        except ClientError as err:  
            logger.error(  
                "Couldn't delete database %s. Here's why: %s: %s", name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise
```

- For API details, see [DeleteDatabase](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an AWS Glue job definition using an AWS SDK

The following code examples show how to delete an AWS Glue job definition and all associated runs.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Deletes the named job.  
/// </summary>  
/// <param name="glueClient">The initialized AWS Glue client.</param>  
/// <param name="jobName">The name of the job to delete.</param>  
/// <returns>A Boolean value indicating whether the AWS Glue job was  
/// deleted successfully.</returns>  
public static async Task<bool> DeleteJobAsync(AmazonGlueClient glueClient,  
string jobName)  
{  
    var jobRequest = new DeleteJobRequest  
    {
```

```
        JobName = jobName,
    };

    var response = await glueClient.DeleteJobAsync(jobRequest);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{jobName} was successfully deleted");
        return true;
    }

    Console.WriteLine($"{jobName} could not be deleted.");
    return false;
}
```

- For API details, see [DeleteJob](#) in *AWS SDK for .NET API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const deleteJob = (jobName) => {
    const client = new GlueClient({ region: DEFAULT_REGION });

    const command = new DeleteJobCommand({
        JobName: jobName,
    });

    return client.send(command);
};
```

- For API details, see [DeleteJob](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
echo "Delete the job.\n";
$glueClient->deleteJob([
    'JobName' => $job['Name'],
]);

public function deleteJob($jobName)
{
    return $this->glueClient->deleteJob([
        'JobName' => $jobName,
    ]);
}
```

- For API details, see [DeleteJob](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def delete_job(self, job_name):  
        """  
        Deletes a job definition. This also deletes data about all runs that are  
        associated with this job definition.  
  
        :param job_name: The name of the job definition to delete.  
        """  
        try:  
            self.glue_client.delete_job(JobName=job_name)  
        except ClientError as err:  
            logger.error(  
                "Couldn't delete job %s. Here's why: %s: %s", job_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise
```

- For API details, see [DeleteJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a table from an AWS Glue Data Catalog database using an AWS SDK

The following code examples show how to delete a table from an AWS Glue Data Catalog database.

### JavaScript

#### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const deleteTable = (databaseName, tableName) => {  
    const client = new GlueClient({ region: DEFAULT_REGION });  
  
    const command = new DeleteTableCommand({  
        DatabaseName: databaseName,  
        Name: tableName,  
    });  
  
    return client.send(command);  
};
```

- For API details, see [DeleteTable in AWS SDK for JavaScript API Reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
echo "Delete the tables.\n";
foreach ($tables['TableList'] as $table) {
    $glueService->deleteTable($table['Name'], $databaseName);
}

public function deleteTable($tableName, $databaseName)
{
    return $this->glueClient->deleteTable([
        'DatabaseName' => $databaseName,
        'Name' => $tableName,
    ]);
}
```

- For API details, see [DeleteTable in AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client

    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                table_name,
                err.response['Error']['Code'],
                err.response['Error']['Message']
            )
            raise
```

- For API details, see [DeleteTable in AWS SDK for Python \(Boto3\) API Reference](#).

## Get an AWS Glue crawler using an AWS SDK

The following code examples show how to get an AWS Glue crawler.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Retrieves information about a specific AWS Glue crawler.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="crawlerName">The name of the crawler.</param>
/// <returns>A Boolean value indicating whether information about
/// the AWS Glue crawler was retrieved successfully.</returns>
public static async Task<bool> GetSpecificCrawlerAsync(AmazonGlueClient
glueClient, string crawlerName)
{
    var crawlerRequest = new GetCrawlerRequest
    {
        Name = crawlerName,
    };

    var response = await glueClient.GetCrawlerAsync(crawlerRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        var databaseName = response.Crawler.DatabaseName;
        Console.WriteLine($"'{crawlerName}' has the database
{databaseName}");
        return true;
    }

    Console.WriteLine($"No information regarding '{crawlerName}' could be
found.");
    return false;
}
```

- For API details, see [GetCrawler in AWS SDK for .NET API Reference](#).

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getSpecificCrawler(GlueClient glueClient, String
crawlerName) {
```

```
try {
    GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
        .name(crawlerName)
        .build();

    GetCrawlerResponse response = glueClient.getcrawler(crawlerRequest);
    Instant createDate = response.crawler().creationTime();

    // Convert the Instant to readable date
    DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

    formatter.format(createDate);
    System.out.println("The create date of the Crawler is " + createDate);

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [GetCrawler in AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getcrawler = (name) => {
    const client = new GlueClient({ region: DEFAULT_REGION });

    const command = new GetCrawlerCommand({
        Name: name,
    });

    return client.send(command);
};
```

- For API details, see [GetCrawler in AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getSpecificCrawler(crawlerName: String?) {
```

```
    val request = GetCrawlerRequest {
        name = crawlerName
    }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getcrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}
```

- For API details, see [GetCrawler in AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
echo "Waiting for crawler";
do {
    $crawler = $glueService->getcrawler($crawlerName);
    echo ".";
    sleep(10);
} while ($crawler['Crawler']['State'] != "READY");
echo "\n";

public function getcrawler($crawlerName)
{
    return $this->customWaiter(function () use ($crawlerName) {
        return $this->glueClient->getcrawler([
            'Name' => $crawlerName,
        ]);
    });
}
```

- For API details, see [GetCrawler in AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client

    def get_crawler(self, name):
        """
```

```
Gets information about a crawler.

:param name: The name of the crawler to look up.
:return: Data about the crawler.
"""
crawler = None
try:
    response = self.glue_client.get_crawler(Name=name)
    crawler = response['Crawler']
except ClientError as err:
    if err.response['Error']['Code'] == 'EntityNotFoundException':
        logger.info("Crawler %s doesn't exist.", name)
    else:
        logger.error(
            "Couldn't get crawler %s. Here's why: %s: %s",
            name,
            err.response['Error']['Code'],
            err.response['Error']
        )
raise
return crawler
```

- For API details, see [GetCrawler in AWS SDK for Python \(Boto3\) API Reference](#).

## Get a database from the AWS Glue Data Catalog using an AWS SDK

The following code examples show how to get a database from the AWS Glue Data Catalog.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Gets information about the database created for this Glue
/// example.
/// </summary>
/// <param name="glueClient">The initialized Glue client.</param>
/// <param name="databaseName">The name of the AWS Glue database.</param>
/// <returns>A Boolean value indicating whether information about
/// the AWS Glue database was retrieved successfully.</returns>
public static async Task<bool> GetSpecificDatabaseAsync(
    AmazonGlueClient glueClient,
    string databaseName)
{
    var databasesRequest = new GetDatabaseRequest
    {
        Name = databaseName,
    };

    var response = await glueClient.GetDatabaseAsync(databasesRequest);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"The Create Time is
{response.Database.CreateTime}");
        return true;
    }
}
```

```
        }

        Console.WriteLine($"No information about {databaseName}.");
        return false;
    }
}
```

- For API details, see [GetDatabase](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {

    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response =
        glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the database is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [GetDatabase](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getDatabase = (name) => {
    const client = new GlueClient({ region: DEFAULT_REGION });

    const command = new GetDatabaseCommand({
```

```
        Name: name,  
    });  
  
    return client.send(command);  
};
```

- For API details, see [GetDatabase](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getSpecificDatabase(databaseName: String?) {  
  
    val request = GetDatabaseRequest {  
        name = databaseName  
    }  
  
    GlueClient { region = "us-east-1" }.use { glueClient ->  
        val response = glueClient.getDatabase(request)  
        val dbDesc = response.database?.description  
        println("The database description is $dbDesc")  
    }  
}
```

- For API details, see [GetDatabase](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$databaseName = "doc-example-database-$unqid";  
  
$database = $glueService->getDatabase($databaseName);  
echo "Found a database named " . $database['Database']['Name'] . "\n";  
  
public function getDatabase(string $databaseName): Result  
{  
    return $this->customWaiter(function () use ($databaseName) {  
        return $this->glueClient->getDatabase([  
            'Name' => $databaseName,  
        ]);  
    });  
}
```

- For API details, see [GetDatabase](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def get_database(self, name):  
        """  
        Gets information about a database in your Data Catalog.  
  
        :param name: The name of the database to look up.  
        :return: Information about the database.  
        """  
        try:  
            response = self.glue_client.get_database(Name=name)  
        except ClientError as err:  
            logger.error(  
                "Couldn't get database %s. Here's why: %s: %s", name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return response['Database']
```

- For API details, see [GetDatabase](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get an AWS Glue job run using an AWS SDK

The following code examples show how to get an AWS Glue job run.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getJobRun = (jobName, jobRunId) => {  
    const client = new GlueClient({ region: DEFAULT_REGION });  
    const command = new GetJobRunCommand({  
        JobName: jobName,  
        RunId: jobRunId,  
    });  
  
    return client.send(command);  
};
```

- For API details, see [GetJobRun in AWS SDK for JavaScript API Reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$jobName = 'test-job-' . $uniqid;

$outputBucketUrl = "s3://$bucketName";
$runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

echo "waiting for job";
do {
    $jobRun = $glueService->getJobRun($jobName, $runId);
    echo ".";
    sleep(10);
} while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
echo "\n";

public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
{
    return $this->glueClient->getJobRun([
        'JobName' => $jobName,
        'RunId' => $runId,
        'PredecessorsIncluded' => $predecessorsIncluded,
    ]);
}
```

- For API details, see [GetJobRun in AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client

    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.
```

```
:param name: The name of the job definition for the run.
:param run_id: The ID of the run.
:return: Information about the run.
"""
try:
    response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
except ClientError as err:
    logger.error(
        "Couldn't get job run %s/%s. Here's why: %s: %s",
        name, run_id,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response['JobRun']
```

- For API details, see [GetJobRun](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get database list from the AWS Glue Data Catalog using an AWS SDK

The following code example shows how to get a list of databases from the AWS Glue Data Catalog.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getDatabases = () => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new GetDatabasesCommand({});

  return client.send(command);
};
```

- For API details, see [GetDatabases](#) in *AWS SDK for JavaScript API Reference*.

## Get a job from the AWS Glue Data Catalog using an AWS SDK

The following code example shows how to get a job from the AWS Glue Data Catalog.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getJob = (jobName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new GetJobCommand({
    JobName: jobName,
```

```
});  
  
    return client.send(command);  
};
```

- For API details, see [GetJob](#) in *AWS SDK for JavaScript API Reference*.

## Get runs of an AWS Glue job using an AWS SDK

The following code examples show how to get runs of an AWS Glue job.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>  
/// Retrieves information about an AWS Glue job.  
///</summary>  
///<param name="glueClient">The initialized AWS Glue client.</param>  
///<param name="jobName">The AWS Glue object for which to retrieve run  
/// information.</param>  
///<returns>A Boolean value indicating whether information about  
/// the AWS Glue job runs was retrieved successfully.</returns>  
public static async Task<bool> GetJobRunsAsync(AmazonGlueClient glueClient,  
string jobName)  
{  
    var runsRequest = new GetJobRunsRequest  
    {  
        JobName = jobName,  
        MaxResults = 20,  
    };  
  
    var response = await glueClient.GetJobRunsAsync(runsRequest);  
    var jobRuns = response.JobRuns;  
  
    if (jobRuns.Count > 0)  
    {  
        foreach (JobRun jobRun in jobRuns)  
        {  
            Console.WriteLine($"Job run state is {jobRun.JobRunState}");  
            Console.WriteLine($"Job run Id is {jobRun.Id}");  
            Console.WriteLine($"The Glue version is {jobRun.GlueVersion}");  
        }  
  
        return true;  
    }  
    else  
    {  
        Console.WriteLine("No jobs found.");  
        return false;  
    }  
}
```

- For API details, see [GetJobRuns](#) in *AWS SDK for .NET API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- For API details, see [GetJobRuns in AWS SDK for JavaScript API Reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$jobName = 'test-job-' . $uniqid;

$jobRuns = $glueService->getJobRuns($jobName);

public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
{
    $arguments = ['JobName' => $jobName];
    if ($maxResults) {
        $arguments['MaxResults'] = $maxResults;
    }
    if ($nextToken) {
        $arguments['NextToken'] = $nextToken;
    }
    return $this->glueClient->getJobRuns($arguments);
}
```

- For API details, see [GetJobRuns in AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
```

```
:param glue_client: A Boto3 Glue client.  
"""  
    self.glue_client = glue_client  
  
def get_job_runs(self, job_name):  
    """  
        Gets information about runs that have been performed for a specific job  
        definition.  
  
        :param job_name: The name of the job definition to look up.  
        :return: The list of job runs.  
    """  
    try:  
        response = self.glue_client.get_job_runs(JobName=job_name)  
    except ClientError as err:  
        logger.error(  
            "Couldn't get job runs for %s. Here's why: %s: %s", job_name,  
            err.response['Error']['Code'], err.response['Error']['Message'])  
        raise  
    else:  
        return response['JobRuns']
```

- For API details, see [GetJobRuns in AWS SDK for Python \(Boto3\) API Reference](#).

## Get tables from a database in the AWS Glue Data Catalog using an AWS SDK

The following code examples show how to get tables from a database in the AWS Glue Data Catalog.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>  
/// Gets the tables used by the database for an AWS Glue crawler.  
///</summary>  
///<param name="glueClient">The initialized AWS Glue client.</param>  
///<param name="dbName">The name of the database.</param>  
///<returns>A Boolean value indicating whether information about  
///the AWS Glue tables was retrieved successfully.</returns>  
public static async Task<bool> GetGlueTablesAsync(  
    AmazonGlueClient glueClient,  
    string dbName)  
{  
    var tableRequest = new GetTablesRequest  
    {  
        DatabaseName = dbName,  
    };  
  
    // Get the list of AWS Glue databases.  
    var response = await glueClient.GetTablesAsync(tableRequest);  
    var tables = response.TableList;  
  
    if (tables.Count > 0)  
    {
```

```
// Displays the list of table names.  
tables.ForEach(table => { Console.WriteLine($"Table name is:  
{table.Name}"); });  
    return true;  
}  
else  
{  
    Console.WriteLine("No tables found.");  
    return false;  
}  
}
```

- For API details, see [GetTables](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getGlueTable(GlueClient glueClient, String dbName, String  
tableName ) {  
  
    try {  
        GetTableRequest tableRequest = GetTableRequest.builder()  
            .databaseName(dbName)  
            .name(tableName)  
            .build();  
  
        GetTableResponse tableResponse = glueClient.getTable(tableRequest);  
        Instant createDate = tableResponse.table().createTime();  
  
        // Convert the Instant to readable date.  
        DateTimeFormatter formatter =  
DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )  
            .withLocale( Locale.US )  
            .withZone( ZoneId.systemDefault() );  
  
        formatter.format( createDate );  
        System.out.println("The create date of the table is " + createDate );  
  
    } catch (GlueException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [GetTables](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getTables = (databaseName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};
```

- For API details, see [GetTables](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$databaseName = "doc-example-database-$uniqid";

$tables = $glueService->getTables($databaseName);

public function getTables($databaseName): Result
{
    return $this->glueClient->getTables([
        'DatabaseName' => $databaseName,
    ]);
}
```

- For API details, see [GetTables](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client

    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.

        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
        """
        try:
```

```
        response = self.glue_client.get_tables(DatabaseName=db_name)
    except ClientError as err:
        logger.error(
            "Couldn't get tables %s. Here's why: %s: %s",
            db_name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return response['TableList']
```

- For API details, see [GetTables](#) in *AWS SDK for Python (Boto3) API Reference*.

## List AWS Glue job definitions using an AWS SDK

The following code examples show how to list AWS Glue job definitions.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const listJobs = () => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new ListJobsCommand({});

  return client.send(command);
};
```

- For API details, see [ListJobs](#) in *AWS SDK for JavaScript API Reference*.

PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$jobs = $glueService->listJobs();
echo "Current jobs:\n";
foreach ($jobs['JobNames'] as $jobsName) {
    echo "{$jobsName}\n";
}

public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
{
    $arguments = [];
    if ($maxResults) {
        $arguments['MaxResults'] = $maxResults;
    }
    if ($nextToken) {
        $arguments['NextToken'] = $nextToken;
    }
    if (!isEmpty($tags)) {
```

```
        $arguments['Tags'] = $tags;
    }
    return $this->glueClient->listJobs($arguments);
}
```

- For API details, see [ListJobs](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client

    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['JobNames']
```

- For API details, see [ListJobs](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start an AWS Glue crawler using an AWS SDK

The following code examples show how to start an AWS Glue crawler.

### .NET

#### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Starts the named AWS Glue crawler.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
```

```
    /// <param name="crawlerName">The name of the crawler to start.</param>
    /// <returns>A Boolean value indicating whether the AWS Glue crawler
    /// was started successfully.</returns>
    public static async Task<bool> StartSpecificCrawlerAsync(AmazonGlueClient
glueClient, string crawlerName)
{
    var crawlerRequest = new StartCrawlerRequest
    {
        Name = crawlerName,
    };

    var response = await glueClient.StartCrawlerAsync(crawlerRequest);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{crawlerName} was successfully started!");
        return true;
    }

    Console.WriteLine($"Could not start AWS Glue crawler, {crawlerName}.");
    return false;
}
```

- For API details, see [StartCrawler](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {

    try {
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.startCrawler(crawlerRequest);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [StartCrawler](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const startCrawler = (name) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- For API details, see [StartCrawler](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun startSpecificCrawler(crawlerName: String?) {

    val request = StartCrawlerRequest {
        name = crawlerName
    }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- For API details, see [StartCrawler](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$crawlerName = "example-crawler-test-" . $uniqid;
$databaseName = "doc-example-database-$uniqid";
$glueService->startCrawler($crawlerName);

public function startCrawler($crawlerName): Result
{
    return $this->glueClient->startCrawler([
        'Name' => $crawlerName,
```

```
        ]);  
    }
```

- For API details, see [StartCrawler](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def start_crawler(self, name):  
        """  
        Starts a crawler. The crawler crawls its configured target and creates  
        metadata that describes the data it finds in the target data source.  
  
        :param name: The name of the crawler to start.  
        """  
        try:  
            self.glue_client.start_crawler(Name=name)  
        except ClientError as err:  
            logger.error(  
                "Couldn't start crawler %s. Here's why: %s: %s", name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise
```

- For API details, see [StartCrawler](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start an AWS Glue job run using an AWS SDK

The following code examples show how to start an AWS Glue job run.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Starts an AWS Glue job.  
/// </summary>  
/// <param name="glueClient">The initialized Glue client.</param>  
/// <param name="jobName">The name of the AWS Glue job to start.</param>
```

```
    /// <returns>A Boolean value indicating whether the AWS Glue job
    /// was started successfully.</returns>
    public static async Task<bool> StartJobAsync(AmazonGlueClient glueClient,
string jobName)
{
    var runRequest = new StartJobRunRequest
    {
        WorkerType = WorkerType.G1X,
        NumberOfWorkers = 10,
        JobName = jobName,
    };

    var response = await glueClient.StartJobRunAsync(runRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"'{jobName}' successfully started. The job run id
is {response.JobRunId}.");
        return true;
    }

    Console.WriteLine($"Could not start {jobName}.");
    return false;
}
```

- For API details, see [StartJobRun](#) in *AWS SDK for .NET API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
      "--output_bucket_url": `s3://${bucketName}/`  

    }
  });

  return client.send(command);
};
```

- For API details, see [StartJobRun](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$jobName = 'test-job-' . $uniqid;  
  
$databaseName = "doc-example-database-$uniqid";  
  
$tables = $glueService->getTables($databaseName);  
  
$outputBucketUrl = "s3://$bucketName";  
$runId = $glueService->startJobRun($jobName, $databaseName, $tables,  
$outputBucketUrl)['JobRunId'];  
  
public function startJobRun($jobName, $databaseName, $tables,  
$outputBucketUrl): Result  
{  
    return $this->glueClient->startJobRun([  
        'JobName' => $jobName,  
        'Arguments' => [  
            'input_database' => $databaseName,  
            'input_table' => $tables['TableList'][0]['Name'],  
            'output_bucket_url' => $outputBucketUrl,  
            '--input_database' => $databaseName,  
            '--input_table' => $tables['TableList'][0]['Name'],  
            '--output_bucket_url' => $outputBucketUrl,  
        ],  
    ]);  
}
```

- For API details, see [StartJobRun](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def start_job_run(self, name, input_database, input_table, output_bucket_name):  
        """  
        Starts a job run. A job run extracts data from the source, transforms it,  
        and loads it to the output bucket.  
  
        :param name: The name of the job definition.  
        :param input_database: The name of the metadata database that contains  
        tables  
                           that describe the source data. This is typically  
        created  
                           by a crawler.  
        :param input_table: The name of the table in the metadata database that  
                           describes the source data.  
        :param output_bucket_name: The S3 bucket where the output is written.  
        :return: The ID of the job run.  
        """
```

```
try:
    # The custom Arguments that are passed to this function are used by the
    # Python ETL script to determine the location of input and output data.
    response = self.glue_client.start_job_run(
        JobName=name,
        Arguments={
            '--input_database': input_database,
            '--input_table': input_table,
            '--output_bucket_url': f's3://{output_bucket_name}/'})
except ClientError as err:
    logger.error(
        "Couldn't start job run %s. Here's why: %s: %s",
        name,
        err.response['Error']['Code'],
        err.response['Error']['Message'])
    raise
else:
    return response['JobRunId']
```

- For API details, see [StartJobRun](#) in [AWS SDK for Python \(Boto3\) API Reference](#).

## Scenarios for AWS Glue using AWS SDKs

The following code examples show how to use AWS Glue with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Get started running AWS Glue crawlers and jobs using an AWS SDK \(p. 817\)](#)

## Get started running AWS Glue crawlers and jobs using an AWS SDK

The following code examples show how to:

- Create and run a crawler that crawls a public Amazon Simple Storage Service (Amazon S3) bucket and generates a metadata database that describes the CSV-formatted data it finds.
- List information about databases and tables in your AWS Glue Data Catalog.
- Create and run a job that extracts CSV data from the source Amazon S3 bucket, transforms it by removing and renaming fields, and loads JSON-formatted output into another Amazon S3 bucket.
- List information about job runs and view some of the transformed data.
- Delete all resources created by the demo.

For more information, see [Tutorial: Getting started with AWS Glue Studio](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps AWS Glue functions that are used in the scenario.

```
namespace Glue_Basics
{
```

```

    ///<summary>
    /// Methods for working the AWS Glue by using the AWS SDK for .NET (v3.7).
    /// </summary>
    public static class GlueMethods
    {

        ///<summary>
        /// Creates a database for use by an AWS Glue crawler.
        /// </summary>
        ///<param name="glueClient">The initialized AWS Glue client.</param>
        ///<param name="dbName">The name of the new database.</param>
        ///<param name="locationUri">The location of scripts that will be
        /// used by the AWS Glue crawler.</param>
        ///<returns>A Boolean value indicating whether the AWS Glue database
        /// was created successfully.</returns>
        public static async Task<bool> CreateDatabaseAsync(AmazonGlueClient
glueClient, string dbName, string locationUri)
        {
            try
            {
                var DataBaseInput = new DatabaseInput
                {
                    Description = "Built with the AWS SDK for .NET (v3)",
                    Name = dbName,
                    LocationUri = locationUri,
                };

                var request = new CreateDatabaseRequest
                {
                    DatabaseInput = DataBaseInput,
                };

                var response = await glueClient.CreateDatabaseAsync(request);
                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    Console.WriteLine("The database was successfully created");
                    return true;
                }
                else
                {
                    Console.WriteLine("Could not create the database.");
                    return false;
                }
            }
            catch (AmazonGlueException ex)
            {
                Console.WriteLine($"Error occurred: '{ex.Message}'");
                return false;
            }
        }
    }

    ///<summary>
    /// Creates an AWS Glue crawler.
    /// </summary>
    ///<param name="glueClient">The initialized AWS Glue client.</param>
    ///<param name="iam">The Amazon Resource Name (ARN) of the IAM role
    /// that is used by the crawler.</param>
    ///<param name="s3Path">The path to the Amazon S3 bucket where
    /// data is stored.</param>
    ///<param name="cron">The name of the CRON job that runs the crawler.</
param>
    ///<param name="dbName">The name of the database.</param>
    ///<param name="crawlerName">The name of the AWS Glue crawler.</param>
    ///<returns>A Boolean value indicating whether the AWS Glue crawler was

```

```

///> created successfully.</returns>
public static async Task<bool> CreateGlueCrawlerAsync(
    AmazonGlueClient glueClient,
    string iam,
    string s3Path,
    string cron,
    string dbName,
    string crawlerName)
{
    var s3Target = new S3Target
    {
        Path = s3Path,
    };

    var targetList = new List<S3Target>
    {
        s3Target,
    };

    var targets = new CrawlerTargets
    {
        S3Targets = targetList,
    };

    var crawlerRequest = new CreateCrawlerRequest
    {
        DatabaseName = dbName,
        Name = crawlerName,
        Description = "Created by the AWS Glue .NET API",
        Targets = targets,
        Role = iam,
        Schedule = cron,
    };

    var response = await glueClient.CreateCrawlerAsync(crawlerRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{crawlerName} was successfully created");
        return true;
    }
    else
    {
        Console.WriteLine($"Could not create {crawlerName}.");
        return false;
    }
}

///> summary>
///> Creates an AWS Glue job.
///> </summary>
///> <param name="glueClient">The initialized AWS Glue client.</param>
///> <param name="jobName">The name of the job to create.</param>
///> <param name="iam">The Amazon Resource Name (ARN) of the IAM role
///> that will be used by the job.</param>
///> <param name="scriptLocation">The location where the script is stored.</param>
///> <returns>A Boolean value indicating whether the AWS Glue job was
///> created successfully.</returns>
public static async Task<bool> CreateJobAsync(AmazonGlueClient glueClient,
string jobName, string iam, string scriptLocation)
{
    var command = new JobCommand
    {
        PythonVersion = "3",
}

```

```
Name = "MyJob1",
    ScriptLocation = scriptLocation,
};

var jobRequest = new CreateJobRequest
{
    Description = "A Job created by using the AWS SDK for .NET",
    GlueVersion = "2.0",
    WorkerType = WorkerType.G1X,
    NumberOfWorkers = 10,
    Name = jobName,
    Role = iam,
    Command = command,
};

var response = await glueClient.CreateJobAsync(jobRequest);

if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
{
    Console.WriteLine($"'{jobName}' was successfully created.");
    return true;
}

Console.WriteLine($"'{jobName}' could not be created.");
return false;
}

/// <summary>
/// Deletes the named AWS Glue crawler.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="crawlerName">The name of the crawler to delete.</param>
/// <returns>A Boolean value indicating whether the AWS Glue crawler was
/// deleted successfully.</returns>
public static async Task<bool> DeleteSpecificCrawlerAsync(AmazonGlueClient
glueClient, string crawlerName)
{
    var deleteCrawlerRequest = new DeleteCrawlerRequest
    {
        Name = crawlerName,
    };

    var response = await
glueClient.DeleteCrawlerAsync(deleteCrawlerRequest);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"'{crawlerName}' was deleted");
        return true;
    }

    Console.WriteLine($"Could not create '{crawlerName}'.");
    return false;
}

/// <summary>
/// Deletes an AWS Glue database.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="databaseName">The name of the database to delete.</param>
/// <returns>A Boolean value indicating whether the AWS Glue database was
/// deleted successfully.</returns>
```

```
        public static async Task<bool> DeleteDatabaseAsync(AmazonGlueClient
glueClient, string databaseName)
{
    var request = new DeleteDatabaseRequest
    {
        Name = databaseName,
    };

    var response = await glueClient.DeleteDatabaseAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{databaseName} was successfully deleted");
        return true;
    }

    Console.WriteLine($"{databaseName} could not be deleted.");
    return false;
}

/// <summary>
/// Deletes the named job.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="jobName">The name of the job to delete.</param>
/// <returns>A Boolean value indicating whether the AWS Glue job was
/// deleted successfully.</returns>
public static async Task<bool> DeleteJobAsync(AmazonGlueClient glueClient,
string jobName)
{
    var jobRequest = new DeleteJobRequest
    {
        JobName = jobName,
    };

    var response = await glueClient.DeleteJobAsync(jobRequest);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{jobName} was successfully deleted");
        return true;
    }

    Console.WriteLine($"{jobName} could not be deleted.");
    return false;
}

/// <summary>
/// Gets a list of AWS Glue jobs.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <returns>A Boolean value indicating whether information about the
/// AWS Glue jobs was retrieved successfully.</returns>
/// <returns>A Boolean value indicating whether information about
/// all AWS Glue jobs was retrieved.</returns>
public static async Task<bool> GetAllJobsAsync(AmazonGlueClient glueClient)
{
    var jobsRequest = new GetJobsRequest
    {
        MaxResults = 10,
    };
}
```

```
        var response = await glueClient.GetJobsAsync(jobsRequest);
        var jobs = response.Jobs;
        if (jobs.Count > 0)
        {
            jobs.ForEach(job => { Console.WriteLine($"The job name is: {job.Name}"); });
            return true;
        }
        else
        {
            Console.WriteLine("Didn't find any jobs.");
            return false;
        }
    }

/// <summary>
/// Gets the tables used by the database for an AWS Glue crawler.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="dbName">The name of the database.</param>
/// <returns>A Boolean value indicating whether information about
/// the AWS Glue tables was retrieved successfully.</returns>
public static async Task<bool> GetGlueTablesAsync(
    AmazonGlueClient glueClient,
    string dbName)
{
    var tableRequest = new GetTablesRequest
    {
        DatabaseName = dbName,
    };

    // Get the list of AWS Glue databases.
    var response = await glueClient.GetTablesAsync(tableRequest);
    var tables = response.TableList;

    if (tables.Count > 0)
    {
        // Displays the list of table names.
        tables.ForEach(table => { Console.WriteLine($"Table name is: {table.Name}"); });
        return true;
    }
    else
    {
        Console.WriteLine("No tables found.");
        return false;
    }
}

/// <summary>
/// Retrieves information about an AWS Glue job.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="jobName">The AWS Glue object for which to retrieve run
/// information.</param>
/// <returns>A Boolean value indicating whether information about
/// the AWS Glue job runs was retrieved successfully.</returns>
public static async Task<bool> GetJobRunsAsync(AmazonGlueClient glueClient,
string jobName)
{
    var runsRequest = new GetJobRunsRequest
{
```

```
        JobName = jobName,
        MaxResults = 20,
    };

    var response = await glueClient.GetJobRunsAsync(runsRequest);
    var jobRuns = response.JobRuns;

    if (jobRuns.Count > 0)
    {
        foreach (JobRun jobRun in jobRuns)
        {
            Console.WriteLine($"Job run state is {jobRun.JobRunState}");
            Console.WriteLine($"Job run Id is {jobRun.Id}");
            Console.WriteLine($"The Glue version is {jobRun.GlueVersion}");
        }

        return true;
    }
    else
    {
        Console.WriteLine("No jobs found.");
        return false;
    }
}

/// <summary>
/// Retrieves information about a specific AWS Glue crawler.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="crawlerName">The name of the crawler.</param>
/// <returns>A Boolean value indicating whether information about
/// the AWS Glue crawler was retrieved successfully.</returns>
public static async Task<bool> GetSpecificCrawlerAsync(AmazonGlueClient
glueClient, string crawlerName)
{
    var crawlerRequest = new GetCrawlerRequest
    {
        Name = crawlerName,
    };

    var response = await glueClient.GetCrawlerAsync(crawlerRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        var databaseName = response.Crawler.DatabaseName;
        Console.WriteLine($"{crawlerName} has the database
{databaseName}");
        return true;
    }

    Console.WriteLine($"No information regarding {crawlerName} could be
found.");
    return false;
}

/// <summary>
/// Gets information about the database created for this Glue
/// example.
/// </summary>
/// <param name="glueClient">The initialized Glue client.</param>
/// <param name="databaseName">The name of the AWS Glue database.</param>
/// <returns>A Boolean value indicating whether information about
/// the AWS Glue database was retrieved successfully.</returns>
```

```
public static async Task<bool> GetSpecificDatabaseAsync(
    AmazonGlueClient glueClient,
    string databaseName)
{
    var databasesRequest = new GetDatabaseRequest
    {
        Name = databaseName,
    };

    var response = await glueClient.GetDatabaseAsync(databasesRequest);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"The Create Time is {response.Database.CreateTime}");
        return true;
    }

    Console.WriteLine($"No information about {databaseName}.");
    return false;
}

/// <summary>
/// Starts an AWS Glue job.
/// </summary>
/// <param name="glueClient">The initialized Glue client.</param>
/// <param name="jobName">The name of the AWS Glue job to start.</param>
/// <returns>A Boolean value indicating whether the AWS Glue job
/// was started successfully.</returns>
public static async Task<bool> StartJobAsync(AmazonGlueClient glueClient,
string jobName)
{
    var runRequest = new StartJobRunRequest
    {
        WorkerType = WorkerType.G1X,
        NumberOfWorkers = 10,
        JobName = jobName,
    };

    var response = await glueClient.StartJobRunAsync(runRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{jobName} successfully started. The job run id
is {response.JobRunId}.");
        return true;
    }

    Console.WriteLine($"Could not start {jobName}.");
    return false;
}

/// <summary>
/// Starts the named AWS Glue crawler.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="crawlerName">The name of the crawler to start.</param>
/// <returns>A Boolean value indicating whether the AWS Glue crawler
/// was started successfully.</returns>
public static async Task<bool> StartSpecificCrawlerAsync(AmazonGlueClient
glueClient, string crawlerName)
{
    var crawlerRequest = new StartCrawlerRequest
```

```
        {
            Name = crawlerName,
        };

        var response = await glueClient.StartCrawlerAsync(crawlerRequest);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"'{crawlerName}' was successfully started!");
            return true;
        }

        Console.WriteLine($"Could not start AWS Glue crawler, {crawlerName}.");
        return false;
    }
}
```

Create a class that runs the scenario.

```
global using Amazon.Glue;
global using Amazon.Glue.Model;
global using Glue_Basics;

// This example uses .NET Core 6 and the AWS SDK for .NET (v3.7)
// Before running the code, set up your development environment,
// including your credentials. For more information, see the
// following topic:
//     https://docs.aws.amazon.com/sdk-for-net/v3/developer-guide/net-dg-config.html
//
// To set up the resources you need, see the following topic:
//     https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
//
// This example performs the following tasks:
//     1. CreateDatabase
//     2. CreateCrawler
//     3. GetCrawler
//     4. StartCrawler
//     5. GetDatabase
//     6. GetTables
//     7. CreateJob
//     8. StartJobRun
//     9. ListJobs
//    10. GetJobRuns
//    11. DeleteJob
//    12. DeleteDatabase
//    13. DeleteCrawler

// Initialize the values that we need for the scenario.
// The Amazon Resource Name (ARN) of the service role used by the crawler.
var iam = "arn:aws:iam::012345678901:role/AWSGlueServiceRole-CrawlerTutorial";

// The path to the Amazon S3 bucket where the comma-delimited file is stored.
var s3Path = "s3://crawler-public-us-east-1/flight/2016/csv";

var cron = "cron(15 12 * * ? *)";

// The name of the database used by the crawler.
var dbName = "example-flights-db";

var crawlerName = "Flight Data Crawler";
```

```
var jobName = "glue-job34";
var scriptLocation = "s3://aws-glue-scripts-012345678901-us-west-1/GlueDemoUser";
var locationUri = "s3://crawler-public-us-east-1/flight/2016/csv/";

var glueClient = new AmazonGlueClient();
await GlueMethods.DeleteDatabaseAsync(glueClient, dbName);

Console.WriteLine("Creating the database and crawler for the AWS Glue example.");
var success = await GlueMethods.CreateDatabaseAsync(glueClient, dbName,
    locationUri);
success = await GlueMethods.CreateGlueCrawlerAsync(glueClient, iam, s3Path, cron,
    dbName, crawlerName);

// Get information about the AWS Glue crawler.
Console.WriteLine("Showing information about the newly created AWS Glue crawler.");
success = await GlueMethods.GetSpecificCrawlerAsync(glueClient, crawlerName);

Console.WriteLine("Starting the new AWS Glue crawler.");
success = await GlueMethods.StartSpecificCrawlerAsync(glueClient, crawlerName);

Console.WriteLine("Displaying information about the database used by the
    crawler.");
success = await GlueMethods.GetSpecificDatabaseAsync(glueClient, dbName);
success = await GlueMethods.GetGlueTablesAsync(glueClient, dbName);

Console.WriteLine("Creating a new AWS Glue job.");
success = await GlueMethods.CreateJobAsync(glueClient, jobName, iam,
    scriptLocation);

Console.WriteLine("Starting the new AWS Glue job.");
success = await GlueMethods.StartJobAsync(glueClient, jobName);

Console.WriteLine("Getting information about the AWS Glue job.");
success = await GlueMethods.GetAllJobsAsync(glueClient);
success = await GlueMethods.GetJobRunsAsync(glueClient, jobName);

Console.WriteLine("Deleting the AWS Glue job used by the example.");
success = await GlueMethods.DeleteJobAsync(glueClient, jobName);

Console.WriteLine("\n*** Waiting 5 MIN for the " + crawlerName + " to stop. ***");
System.Threading.Thread.Sleep(300000);

Console.WriteLine("Clean up the resources created for the example.");
success = await GlueMethods.DeleteDatabaseAsync(glueClient, dbName);
success = await GlueMethods.DeleteSpecificCrawlerAsync(glueClient, crawlerName);

Console.WriteLine("Successfully completed the AWS Glue Scenario ");
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.

- [CreateCrawler](#)
- [CreateJob](#)
- [DeleteCrawler](#)
- [DeleteDatabase](#)
- [DeleteJob](#)
- [DeleteTable](#)
- [GetCrawler](#)
- [GetDatabase](#)
- [GetDatabases](#)
- [GetJob](#)

- [GetJobRun](#)
- [GetJobRuns](#)
- [GetTables](#)
- [ListJobs](#)
- [StartCrawler](#)
- [StartJobRun](#)

Java

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
 * Before running this Java V2 code example, set up your development environment,  
 * including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * To set up the resources, see this documentation topic:  
 *  
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html  
 *  
 * This example performs the following tasks:  
 *  
 * 1. Create a database.  
 * 2. Create a crawler.  
 * 3. Get a crawler.  
 * 4. Start a crawler.  
 * 5. Get a database.  
 * 6. Get tables.  
 * 7. Create a job.  
 * 8. Start a job run.  
 * 9. List all jobs.  
 * 10. Get job runs.  
 * 11. Delete a job.  
 * 12. Delete a database.  
 * 13. Delete a crawler.  
 */  
  
public class GlueScenario {  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    public static void main(String[] args) throws InterruptedException {  
  
        final String usage = "\n" +  
            "Usage:\n" +  
            "  <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName> \n\n" +  
            "Where:\n" +  
            "  iam - The ARN of the IAM role that has AWS Glue and S3  
permissions. \n" +  
            "  s3Path - The Amazon Simple Storage Service (Amazon S3) target that  
contains data (for example, CSV data).\n" +  
            "  cron - A cron expression used to specify the schedule (i.e.,  
cron(15 12 * * ? *)\n" +  
            "  dbName - The database name. \n" +
```

```
"      crawlerName - The name of the crawler. \n" +
"      jobName - The name you assign to this job definition."+
"      scriptLocation - The Amazon S3 path to a script that runs a job."
+
"      locationUri - The location of the database" ;

if (args.length != 8) {
    System.out.println(usage);
    System.exit(1);
}

String iam = args[0];
String s3Path = args[1];
String cron = args[2];
String dbName = args[3];
String crawlerName = args[4];
String jobName = args[5];
String scriptLocation = args[6];
String locationUri = args[7];

Region region = Region.US_EAST_1;
GlueClient glueClient = GlueClient.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();
System.out.println(DASHES);
System.out.println("Welcome to the AWS Glue scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create a database.");
createDatabase(glueClient, dbName, locationUri);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a crawler.");
createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get a crawler.");
getSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Start a crawler.");
startSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a database.");
getSpecificDatabase(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Get tables.");
getGlueTables(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create a job.");
createJob(glueClient, jobName, iam, scriptLocation);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Start a Job run.");
```

```
        startJob(glueClient, jobName);
System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
getAllJobs(glueClient);
System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
getJobRuns(glueClient, jobName);
System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
deleteJob(glueClient, jobName);
System.out.println("*** Wait 5 MIN for the "+crawlerName +" to stop");
TimeUnit.MINUTES.sleep(5);
System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
deleteDatabase(glueClient, dbName);
System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
deleteSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }

    public static void createDatabase(GlueClient glueClient, String dbName, String
locationUri ) {

    try {
        DatabaseInput input = DatabaseInput.builder()
            .description("Built with the AWS SDK for Java V2")
            .name(dbName)
            .locationUri(locationUri)
            .build();

        CreateDatabaseRequest request = CreateDatabaseRequest.builder()
            .databaseInput(input)
            .build();

        glueClient.createDatabase(request);
        System.out.println("The database was successfully created");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

    public static void createGlueCrawler(GlueClient glueClient,
String iam,
String s3Path,
String cron,
String dbName,
String crawlerName) {

    try {
```

```
S3Target s3Target = S3Target.builder()  
    .path(s3Path)  
    .build();  
  
List<S3Target> targetList = new ArrayList<>();  
targetList.add(s3Target);  
CrawlerTargets targets = CrawlerTargets.builder()  
    .s3Targets(targetList)  
    .build();  
  
CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()  
    .databaseName(dbName)  
    .name(crawlerName)  
    .description("Created by the AWS Glue Java API")  
    .targets(targets)  
    .role(iam)  
    .schedule(cron)  
    .build();  
  
glueClient.createCrawler(crawlerRequest);  
System.out.println(crawlerName +" was successfully created");  
  
} catch (GlueException e) {  
    System.err.println(e.awsErrorDetails().errorMessage());  
    System.exit(1);  
}  
}  
  
}  
  
public static void getSpecificCrawler(GlueClient glueClient, String  
crawlerName) {  
  
    try {  
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()  
            .name(crawlerName)  
            .build();  
  
        GetCrawlerResponse response = glueClient.getcrawler(crawlerRequest);  
        Instant createDate = response.crawler().creationTime();  
        DateTimeFormatter formatter =  
DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )  
            .withLocale( Locale.US )  
            .withZone( ZoneId.systemDefault() );  
  
        formatter.format( createDate );  
        System.out.println("The create date of the Crawler is " + createDate );  
    } catch (GlueException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}  
  
}  
  
public static void startSpecificCrawler(GlueClient glueClient, String  
crawlerName) {  
  
    try {  
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()  
            .name(crawlerName)  
            .build();  
  
        glueClient.startCrawler(crawlerRequest);  
        System.out.println(crawlerName +" was successfully started!");  
    } catch (GlueException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

```
        }

    public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {

    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response =
glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime( TextStyle.SHORT )
            .withLocale( Locale.US )
            .withZone( ZoneId.systemDefault() );

        formatter.format( createDate );
        System.out.println("The create date of the database is " +
createDate );

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getGlueTables(GlueClient glueClient, String dbName){

try {
    GetTablesRequest tableRequest = GetTablesRequest.builder()
        .databaseName(dbName)
        .build();

    GetTablesResponse response = glueClient.getTables(tableRequest);
    List<Table> tables = response.tableList();
    for (Table table: tables) {
        System.out.println("Table name is: "+table.name());
    }

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void startJob(GlueClient glueClient, String jobName) {

try {
    StartJobRunRequest runRequest = StartJobRunRequest.builder()
        .workerType(WorkerType.G_1_X)
        .numberOfWorkers(10)
        .jobName(jobName)
        .build();

    StartJobRunResponse response = glueClient.startJobRun(runRequest);
    System.out.println("The request Id of the job is "+
response.responseMetadata().requestId());

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

```
        }

    public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {

    try {
        JobCommand command = JobCommand.builder()
            .pythonVersion("3")
            .name("MyJob1")
            .scriptLocation(scriptLocation)
            .build();

        CreateJobRequest jobRequest = CreateJobRequest.builder()
            .description("A Job created by using the AWS SDK for Java V2")
            .glueVersion("2.0")
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .name(jobName)
            .role(iam)
            .command(command)
            .build();

        glueClient.createJob(jobRequest);
        System.out.println(jobName + " was successfully created.");
    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getAllJobs(GlueClient glueClient) {

    try {
        GetJobsRequest jobsRequest = GetJobsRequest.builder()
            .maxResults(10)
            .build();

        GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
        List<Job> jobs = jobsResponse.jobs();
        for (Job job: jobs) {
            System.out.println("Job name is : "+job.name());
            System.out.println("The job worker type is :
"+job.workerType().name());
        }
    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getJobRuns(GlueClient glueClient, String jobName) {

    try {
        GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
            .jobName(jobName)
            .maxResults(20)
            .build();

        GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
        List<JobRun> jobRuns = response.jobRuns();
        for (JobRun jobRun: jobRuns) {
            System.out.println("Job run state is
"+jobRun.jobRunState().name());
    }
}
```

```
        System.out.println("Job run Id is "+jobRun.id());
        System.out.println("The Glue version is "+jobRun.glueVersion());
    }

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

public static void deleteJob(GlueClient glueClient, String jobName) {

    try {
        DeleteJobRequest jobRequest = DeleteJobRequest.builder()
            .jobName(jobName)
            .build();

        glueClient.deleteJob(jobRequest);
        System.out.println(jobName +" was successfully deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteDatabase(GlueClient glueClient, String databaseName) {

    try {
        DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
            .name(databaseName)
            .build();

        glueClient.deleteDatabase(request);
        System.out.println(databaseName +" was successfully deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteSpecificCrawler(GlueClient glueClient, String crawlerName) {

    try {
        DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.deleteCrawler(deleteCrawlerRequest);
        System.out.println(crawlerName +" was deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [CreateCrawler](#)

- [CreateJob](#)
- [DeleteCrawler](#)
- [DeleteDatabase](#)
- [DeleteJob](#)
- [DeleteTable](#)
- [GetCrawler](#)
- [GetDatabase](#)
- [GetDatabases](#)
- [GetJob](#)
- [GetJobRun](#)
- [GetJobRuns](#)
- [GetTables](#)
- [ListJobs](#)
- [StartCrawler](#)
- [StartJobRun](#)

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create and run a crawler that crawls a public Amazon Simple Storage Service (Amazon S3) bucket and generates a metadata database that describes the CSV-formatted data it finds.

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: [
      S3Targets: [{ Path: s3TargetPath }],
    ],
  });

  return client.send(command);
};

const getCrawler = (name) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const startCrawler = (name) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new StartCrawlerCommand({
    Name: name,
```

```
});

    return client.send(command);
};

const crawlerExists = async ({ getcrawler }, crawlerName) => {
    try {
        await getcrawler(crawlerName);
        return true;
    } catch {
        return false;
    }
};

const makeCreateCrawlerStep = (actions) => async (context) => {
    if (await crawlerExists(actions, process.env.CRAWLER_NAME)) {
        log("Crawler already exists. Skipping creation.");
    } else {
        await actions.createcrawler(
            process.env.CRAWLER_NAME,
            process.env.ROLE_NAME,
            process.env.DATABASE_NAME,
            process.env.TABLE_PREFIX,
            process.env.S3_TARGET_PATH
        );

        log("Crawler created successfully.", { type: "success" });
    }

    return { ...context };
};

const waitForCrawler = async (getcrawler, crawlerName) => {
    const waitTimeInSeconds = 30;
    const { Crawler } = await getcrawler(crawlerName);

    if (!Crawler) {
        throw new Error(`Crawler with name ${crawlerName} not found.`);
    }

    if (Crawler.State === "READY") {
        return;
    }

    log(`Crawler is ${Crawler.State}. Waiting ${waitTimeInSeconds} seconds...`);
    await wait(waitTimeInSeconds);
    return waitForCrawler(getcrawler, crawlerName);
};

const makeStartCrawlerStep =
    ({ startCrawler, getcrawler }) => {
        async (context) => {
            log("Starting crawler.");
            await startCrawler(process.env.CRAWLER_NAME);
            log("Crawler started.", { type: "success" });

            log("Waiting for crawler to finish running. This can take a while.");
            await waitForCrawler(getcrawler, process.env.CRAWLER_NAME);
            log("Crawler ready.", { type: "success" });

            return { ...context };
        };
    };
}
```

---

List information about databases and tables in your AWS Glue Data Catalog.

```

const getDatabase = (name) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};

const getTables = (databaseName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};

const makeGetDatabaseStep =
  ({ getDatabase }) =>
  async (context) => {
    const {
      Database: { Name },
    } = await getDatabase(process.env.DATABASE_NAME);
    log(`Database: ${Name}`);
    return { ...context };
  };

const makeGetTablesStep =
  ({ getTables }) =>
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME);
    log("Tables:");
    log(TableList.map((table) => `  • ${table.Name}\n`));
    return { ...context };
  };

```

Create and run a job that extracts CSV data from the source Amazon S3 bucket, transforms it by removing and renaming fields, and loads JSON-formatted output into another Amazon S3 bucket.

```

const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};

const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

```

```

const command = new StartJobRunCommand({
  JobName: jobName,
  Arguments: {
    "--input_database": dbName,
    "--input_table": tableName,
    "--output_bucket_url": `s3://${bucketName}/`  

  }
});

return client.send(command);
};

const makeCreateJobStep =
  ({ createJob }) =>
  async (context) => {
    log("Creating Job.");
    await createJob(
      process.env.JOB_NAME,
      process.env.ROLE_NAME,
      process.env.BUCKET_NAME,
      process.env.PYTHON_SCRIPT_KEY
    );
    log("Job created.", { type: "success" });

    return { ...context };
};

const waitForJobRun = async (getJobRun, jobName, jobRunId) => {
  const waitTimeInSeconds = 30;
  const { JobRun } = await getJobRun(jobName, jobRunId);

  if (!JobRun) {
    throw new Error(`Job run with id ${jobRunId} not found.`);
  }

  switch (JobRun.JobRunState) {
    case "FAILED":
    case "STOPPED":
    case "TIMEOUT":
    case "STOPPED":
      throw new Error(
        `Job ${JobRun.JobRunState}. Error: ${JobRun.ErrorMessage}`
      );
    case "RUNNING":
      break;
    case "SUCCEEDED":
      return;
    default:
      throw new Error(`Unknown job run state: ${JobRun.JobRunState}`);
  }

  log(
    `Job ${JobRun.JobRunState}. Waiting ${waitTimeInSeconds} more seconds...`  

  );
  await wait(waitTimeInSeconds);
  return waitForJobRun(getJobRun, jobName, jobRunId);
};

const promptToOpen = async (context) => {
  const { shouldOpen } = await context.prompter.prompt({
    name: "shouldOpen",
    type: "confirm",
    message: "Open the output bucket in your browser?",
  });
}

if (shouldOpen) {

```

```

        return open(
            `https://s3.console.aws.amazon.com/s3/buckets/${process.env.BUCKET_NAME}?
region=${DEFAULT_REGION}&tab=objects to view the output.`);
    );
};

const makeStartJobRunStep =
    ({ startJobRun, getJobRun }) =>
    async (context) => {
        log("Starting job.");
        const { JobRunId } = await startJobRun(
            process.env.JOB_NAME,
            process.env.DATABASE_NAME,
            process.env.TABLE_NAME,
            process.env.BUCKET_NAME
        );
        log("Job started.", { type: "success" });

        log("Waiting for job to finish running. This can take a while.");
        await waitForJobRun(getJobRun, process.env.JOB_NAME, JobRunId);
        log("Job run succeeded.", { type: "success" });

        await promptToOpen(context);

        return { ...context };
};

```

List information about job runs and view some of the transformed data.

```

const getJobRuns = (jobName) => {
    const client = new GlueClient({ region: DEFAULT_REGION });
    const command = new GetJobRunsCommand({
        JobName: jobName,
    });

    return client.send(command);
};

const getJobRun = (jobName, jobRunId) => {
    const client = new GlueClient({ region: DEFAULT_REGION });
    const command = new GetJobRunCommand({
        JobName: jobName,
        RunId: jobRunId,
    });

    return client.send(command);
};

const logJobRunDetails = async (getJobRun, jobName, jobRunId) => {
    const { JobRun } = await getJobRun(jobName, jobRunId);
    log(JobRun, { type: "object" });
};

const makePickJobRunStep =
    ({ getJobRuns, getJobRun }) =>
    async (context) => {
        if (context.selectedJobName) {
            const { JobRuns } = await getJobRuns(context.selectedJobName);

            const { jobRunId } = await context.promoter.prompt({
                name: "jobRunId",
                type: "list",
                message: "Select a job run to see details.",
            });
        }
    };

```

```
        choices: JobRuns.map((run) => run.Id),
    });

    logJobRunDetails(getJobRun, context.selectedJobName, jobRunId);
}

return { ...context };
};
```

Delete all resources created by the demo.

```
const deleteJob = (jobName) => {
    const client = new GlueClient({ region: DEFAULT_REGION });

    const command = new DeleteJobCommand({
        JobName: jobName,
    });

    return client.send(command);
};

const deleteTable = (databaseName, tableName) => {
    const client = new GlueClient({ region: DEFAULT_REGION });

    const command = new DeleteTableCommand({
        DatabaseName: databaseName,
        Name: tableName,
    });

    return client.send(command);
};

const deleteDatabase = (databaseName) => {
    const client = new GlueClient({ region: DEFAULT_REGION });

    const command = new DeleteDatabaseCommand({
        Name: databaseName,
    });

    return client.send(command);
};

const deleteCrawler = (crawlerName) => {
    const client = new GlueClient({ region: DEFAULT_REGION });

    const command = new DeleteCrawlerCommand({
        Name: crawlerName,
    });

    return client.send(command);
};

const handleDeleteJobs = async (deleteJobFn, jobNames, context) => {
    const { selectedJobNames } = await context.prompter.prompt({
        name: "selectedJobNames",
        type: "checkbox",
        message: "Let's clean up jobs. Select jobs to delete.",
        choices: jobNames,
    });

    if (selectedJobNames.length === 0) {
        log("No jobs selected.");
    } else {
        log("Deleting jobs.");
    }
};
```

```
        await Promise.all(
          selectedJobNames.map((n) => deleteJobFn(n).catch(console.error))
        );
        log("Jobs deleted.", { type: "success" });
      }
    );

const makeCleanUpJobsStep =
  ({ listJobs, deleteJob }) =>
  async (context) => {
    const { JobNames } = await listJobs();
    if (JobNames.length > 0) {
      await handleDeleteJobs(deleteJob, JobNames, context);
    }

    return { ...context };
};

const deleteTables = (deleteTable, databaseName, tableNames) =>
Promise.all(
  tableNames.map((tableName) =>
    deleteTable(databaseName, tableName).catch(console.error)
  )
);

const makeCleanUpTablesStep =
  ({ getTables, deleteTable }) =>
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME).catch(
      () => ({ TableList: null })
    );

    if (TableList && TableList.length > 0) {
      const { tableNames } = await context.prompter.prompt({
        name: "tableNames",
        type: "checkbox",
        message: "Let's clean up tables. Select tables to delete.",
        choices: TableList.map((t) => t.Name),
      });

      if (tableNames.length === 0) {
        log("No tables selected.");
      } else {
        log("Deleting tables.");
        await deleteTables(deleteTable, process.env.DATABASE_NAME, tableNames);
        log("Tables deleted.", { type: "success" });
      }
    }

    return { ...context };
};

const deleteDatabases = (deleteDatabase, databaseNames) =>
Promise.all(
  databaseNames.map((dbName) => deleteDatabase(dbName).catch(console.error))
);

const makeCleanUpDatabasesStep =
  ({ getDatabases, deleteDatabase }) =>
  async (context) => {
    const { DatabaseList } = await getDatabases();

    if (DatabaseList.length > 0) {
      const { dbNames } = await context.prompter.prompt({
        name: "dbNames",
        type: "checkbox",
      });
    }
  }
};
```

```
    message: "Let's clean up databases. Select databases to delete.",
    choices: DatabaseList.map((db) => db.Name),
  });

  if (dbNames.length === 0) {
    log("No databases selected.");
  } else {
    log("Deleting databases.");
    await deleteDatabases(deleteDatabase, dbNames);
    log("Databases deleted.", { type: "success" });
  }
}

return { ...context };
};

const cleanUpCrawlerStep = async (context) => {
  log(`Deleting crawler.`);

  try {
    await deleteCrawler(process.env.CRAWLER_NAME);
    log("Crawler deleted.", { type: "success" });
  } catch (err) {
    if (err.name === "EntityNotFoundException") {
      log(`Crawler is already deleted.`);
    } else {
      throw err;
    }
  }

  return { ...context };
};
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.
  - [CreateCrawler](#)
  - [CreateJob](#)
  - [DeleteCrawler](#)
  - [DeleteDatabase](#)
  - [DeleteJob](#)
  - [DeleteTable](#)
  - [GetCrawler](#)
  - [GetDatabase](#)
  - [GetDatabases](#)
  - [GetJob](#)
  - [GetJobRun](#)
  - [GetJobRuns](#)
  - [GetTables](#)
  - [ListJobs](#)
  - [StartCrawler](#)
  - [StartJobRun](#)

Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun main(args: Array<String>) {

    val usage = """
        Usage:
            <iام> <s3Path> <cron> <dbName> <crawlerName> <jobName> <scriptLocation>
            <locationUri>

        Where:
            iam - The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that has AWS Glue and Amazon Simple Storage Service (Amazon S3) permissions.
            s3Path - The Amazon Simple Storage Service (Amazon S3) target that contains data (for example, CSV data).
            cron - A cron expression used to specify the schedule (for example, cron(15 12 * * ? *)).
            dbName - The database name.
            crawlerName - The name of the crawler.
            jobName - The name you assign to this job definition.
            scriptLocation - Specifies the Amazon S3 path to a script that runs a job.
            locationUri - Specifies the location of the database
        """

    if (args.size != 8) {
        println(usage)
        exitProcess(1)
    }

    val iam = args[0]
    val s3Path = args[1]
    val cron = args[2]
    val dbName = args[3]
    val crawlerName = args[4]
    val jobName = args[5]
    val scriptLocation = args[6]
    val locationUri = args[7]

    println("About to start the AWS Glue Scenario")
    createDatabase(dbName, locationUri)
    createCrawler(iam, s3Path, cron, dbName, crawlerName)
    getCrawler(crawlerName)
    startCrawler(crawlerName)
    getDatabase(dbName)
    getGlueTables(dbName)
    createJob(jobName, iam, scriptLocation)
    startJob(jobName)
    getJobs()
    getJobRuns(jobName)
    deleteJob(jobName)
    println("*** Wait for 5 MIN so the $crawlerName is ready to be deleted")
    TimeUnit.MINUTES.sleep(5)
    deleteMyDatabase(dbName)
}
```

```
        deleteCrawler(crawlerName)
    }

    suspend fun createDatabase(dbName: String?, locationUriVal: String?) {

        val input = DatabaseInput {
            description = "Built with the AWS SDK for Kotlin"
            name = dbName
            locationUri = locationUriVal
        }

        val request = CreateDatabaseRequest {
            databaseInput = input
        }

        GlueClient { region = "us-east-1" }.use { glueClient ->
            glueClient.createDatabase(request)
            println("The database was successfully created")
        }
    }

    suspend fun createCrawler(iam: String?, s3Path: String?, cron: String?, dbName: String?, crawlerName: String) {

        val s3Target = S3Target {
            path = s3Path
        }

        val targetList = ArrayList<S3Target>()
        targetList.add(s3Target)

        val target0b = CrawlerTargets {
            s3Targets = targetList
        }

        val crawlerRequest = CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Java API"
            targets = target0b
            role = iam
            schedule = cron
        }

        GlueClient { region = "us-east-1" }.use { glueClient ->
            glueClient.createCrawler(crawlerRequest)
            println("$crawlerName was successfully created")
        }
    }

    suspend fun getCrawler(crawlerName: String?) {

        val request = GetCrawlerRequest {
            name = crawlerName
        }

        GlueClient { region = "us-east-1" }.use { glueClient ->
            val response = glueClient.getcrawler(request)
            val role = response.crawler?.role
            println("The role associated with this crawler is $role")
        }
    }

    suspend fun startCrawler(crawlerName: String) {

        val crawlerRequest = StartCrawlerRequest {
```

```
        name = crawlerName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.startCrawler(crawlerRequest)
        println("$crawlerName was successfully started.")
    }
}

suspend fun getDatabase(databaseName: String?) {

    val request = GetDatabaseRequest {
        name = databaseName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}

suspend fun getGlueTables(dbName: String?) {

    val tableRequest = GetTablesRequest {
        databaseName = dbName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getTables(tableRequest)
        response.tableList?.forEach { tableName ->
            println("Table name is ${tableName.name}")
        }
    }
}

suspend fun startJob(jobNameVal: String?) {

    val runRequest = StartJobRunRequest {
        workerType = WorkerType.G1X
        numberOfWorkers = 10
        jobName = jobNameVal
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.startJobRun(runRequest)
        println("The job run Id is ${response.jobRunId}")
    }
}

suspend fun createJob(jobName: String, iam: String?, scriptLocationVal: String?) {

    val commandOb = JobCommand {
        pythonVersion = "3"
        name = "MyJob1"
        scriptLocation = scriptLocationVal
    }

    val jobRequest = CreateJobRequest {
        description = "A Job created by using the AWS SDK for Java V2"
        glueVersion = "2.0"
        workerType = WorkerType.G1X
        numberOfWorkers = 10
        name = jobName
        role = iam
        command = commandOb
    }
}
```

```
        }

        GlueClient { region = "us-east-1" }.use { glueClient ->
            glueClient.createJob(jobRequest)
            println("$jobName was successfully created.")
        }
    }

suspend fun getJobs() {

    val request = GetJobsRequest {
        maxResults = 10
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobs(request)
        response.jobs?.forEach { job ->
            println("Job name is ${job.name}")
        }
    }
}

suspend fun getJobRuns(jobNameVal: String?) {

    val request = GetJobRunsRequest {
        jobName = jobNameVal
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobRuns(request)
        response.jobRuns?.forEach { job ->
            println("Job name is ${job.jobName}")
        }
    }
}

suspend fun deleteJob(jobNameVal: String) {

    val jobRequest = DeleteJobRequest {
        jobName = jobNameVal
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteJob(jobRequest)
        println("$jobNameVal was successfully deleted")
    }
}

suspend fun deleteMyDatabase(databaseName: String) {

    val request = DeleteDatabaseRequest {
        name = databaseName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteDatabase(request)
        println("$databaseName was successfully deleted")
    }
}

suspend fun deleteCrawler(crawlerName: String) {

    val request = DeleteCrawlerRequest {
        name = crawlerName
    }
    GlueClient { region = "us-east-1" }.use { glueClient ->
```

```
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - [CreateCrawler](#)
  - [CreateJob](#)
  - [DeleteCrawler](#)
  - [DeleteDatabase](#)
  - [DeleteJob](#)
  - [DeleteTable](#)
  - [GetCrawler](#)
  - [GetDatabase](#)
  - [GetDatabases](#)
  - [GetJob](#)
  - [GetJobRun](#)
  - [GetJobRuns](#)
  - [GetTables](#)
  - [ListJobs](#)
  - [StartCrawler](#)
  - [StartJobRun](#)

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace Glue;

use Aws\Glue\GlueClient;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use GuzzleHttp\Psr7\Stream;
use Iam\IamService;

class GettingStartedWithGlue
{
    public function run()
    {
        echo("-----\n");
        print("Welcome to the Amazon Glue getting started demo using PHP!\n");
        echo("-----\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $glueClient = new GlueClient($clientArgs);
```

```
$glueService = new GlueService($glueClient);
$iamService = new IamService();
$crawlerName = "example-crawler-test-" . $uniqid;

AWSServiceClass::$waitTime = 5;
AWSServiceClass::$maxWaitAttempts = 20;

$role = $iamService->getRole("AWSGlueServiceRole-DocExample");

$databaseName = "doc-example-database-$uniqid";
$path = 's3://crawler-public-us-east-1/flight/2016/csv';
$glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);
$glueService->startCrawler($crawlerName);

echo "Waiting for crawler";
do {
    $crawler = $glueService->getCrawler($crawlerName);
    echo ".";
    sleep(10);
} while ($crawler['Crawler']['State'] != "READY");
echo "\n";

$database = $glueService->getDatabase($databaseName);
echo "Found a database named " . $database['Database']['Name'] . "\n";

//Upload job script
$s3client = new S3Client($clientArgs);
$bucketName = "test-glue-bucket-" . $uniqid;
$s3client->createBucket([
    'Bucket' => $bucketName,
    'CreateBucketConfiguration' => ['LocationConstraint' => 'us-west-2'],
]);

$s3client->putObject([
    'Bucket' => $bucketName,
    'Key' => 'run_job.py',
    'SourceFile' => 'glue/flight_etl_job_script.py'
]);
$s3client->putObject([
    'Bucket' => $bucketName,
    'Key' => 'setup_scenario_getting_started.yaml',
    'SourceFile' => 'glue/setup_scenario_getting_started.yaml'
]);

$tables = $glueService->getTables($databaseName);

$jobName = 'test-job-' . $uniqid;
$scriptLocation = "s3://$bucketName/run_job.py";
$job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

$outputBucketUrl = "s3://$bucketName";
$runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

echo "waiting for job";
do {
    $jobRun = $glueService->getJobRun($jobName, $runId);
    echo ".";
    sleep(10);
} while (!array_intersect([$jobRun['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
echo "\n";

$jobRuns = $glueService->getJobRuns($jobName);
```

```
$objects = $s3client->listObjects([
    'Bucket' => $bucketName,
])['Contents'];

foreach ($objects as $object) {
    echo $object['Key'] . "\n";
}

echo "Downloading " . $objects[2]['Key'] . "\n";
/** @var Stream $downloadObject */
$downloadObject = $s3client->getObject([
    'Bucket' => $bucketName,
    'Key' => $objects[2]['Key'],
])['Body']->getContents();
echo "Here is the first 1000 characters in the object.";
echo substr($downloadObject, 0, 1000);

$jobs = $glueService->listJobs();
echo "Current jobs:\n";
foreach ($jobs['JobNames'] as $jobsName) {
    echo "{$jobsName}\n";
}

echo "Delete the job.\n";
$glueClient->deleteJob([
    'JobName' => $job['Name'],
]);

echo "Delete the tables.\n";
foreach ($tables['TableList'] as $table) {
    $glueService->deleteTable($table['Name'], $databaseName);
}

echo "Delete the databases.\n";
$glueClient->deleteDatabase([
    'Name' => $databaseName,
]);

echo "Delete the crawler.\n";
$glueClient->deleteCrawler([
    'Name' => $crawlerName,
]);

$deleteObjects = $s3client->listObjectsV2([
    'Bucket' => $bucketName,
]);
echo "Delete all objects in the bucket.\n";
$deleteObjects = $s3client->deleteObjects([
    'Bucket' => $bucketName,
    'Delete' => [
        'Objects' => $deleteObjects['Contents'],
    ]
]);
echo "Delete the bucket.\n";
$s3client->deleteBucket(['Bucket' => $bucketName]);

echo "This job was brought to you by the number $unqid\n";
}

namespace Glue;

use Aws\Glue\GlueClient;
use Aws\Result;
```

```
use function PHPUnit\Framework\isEmpty;

class GlueService extends \AwsUtilities\AWSServiceClass
{
    protected GlueClient $glueClient;

    public function __construct($glueClient)
    {
        $this->glueClient = $glueClient;
    }

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }

    public function createCrawler($crawlerName, $role, $databaseName, $path): Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                    [
                        [
                            'Path' => $path,
                        ]
                    ],
                ],
            ]);
        });
    }

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([

```

```

        'Name' => $jobName,
        'Role' => $role,
        'Command' => [
            'Name' => 'glueetl',
            'ScriptLocation' => $scriptLocation,
            'PythonVersion' => $pythonVersion,
        ],
        'GlueVersion' => $glueVersion,
    ]);
}

public function startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl): Result
{
    return $this->glueClient->startJobRun([
        'JobName' => $jobName,
        'Arguments' => [
            'input_database' => $databaseName,
            'input_table' => $tables['TableList'][0]['Name'],
            'output_bucket_url' => $outputBucketUrl,
            '--input_database' => $databaseName,
            '--input_table' => $tables['TableList'][0]['Name'],
            '--output_bucket_url' => $outputBucketUrl,
        ],
    ]);
}

public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
{
    $arguments = [];
    if ($maxResults) {
        $arguments['MaxResults'] = $maxResults;
    }
    if ($nextToken) {
        $arguments['NextToken'] = $nextToken;
    }
    if (!isEmpty($tags)) {
        $arguments['Tags'] = $tags;
    }
    return $this->glueClient->listJobs($arguments);
}

public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
{
    $arguments = ['JobName' => $jobName];
    if ($maxResults) {
        $arguments['MaxResults'] = $maxResults;
    }
    if ($nextToken) {
        $arguments['NextToken'] = $nextToken;
    }
    return $this->glueClient->getJobRuns($arguments);
}

public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
{
    return $this->glueClient->getJobRun([
        'JobName' => $jobName,
        'RunId' => $runId,
        'PredecessorsIncluded' => $predecessorsIncluded,
    ]);
}

public function deleteJob($jobName)

```

```
{  
    return $this->glueClient->deleteJob([  
        'JobName' => $jobName,  
    ]);  
}  
  
public function deleteTable($tableName, $databaseName)  
{  
    return $this->glueClient->deleteTable([  
        'DatabaseName' => $databaseName,  
        'Name' => $tableName,  
    ]);  
}  
  
public function deleteDatabase($databaseName)  
{  
    return $this->glueClient->deleteDatabase([  
        'Name' => $databaseName,  
    ]);  
}  
  
public function deleteCrawler($crawlerName)  
{  
    return $this->glueClient->deleteCrawler([  
        'Name' => $crawlerName,  
    ]);  
}  
}
```

- For API details, see the following topics in *AWS SDK for PHP API Reference*.
  - [CreateCrawler](#)
  - [CreateJob](#)
  - [DeleteCrawler](#)
  - [DeleteDatabase](#)
  - [DeleteJob](#)
  - [DeleteTable](#)
  - [GetCrawler](#)
  - [GetDatabase](#)
  - [GetDatabases](#)
  - [GetJob](#)
  - [GetJobRun](#)
  - [GetJobRuns](#)
  - [GetTables](#)
  - [ListJobs](#)
  - [StartCrawler](#)
  - [StartJobRun](#)

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps AWS Glue functions used in the scenario.

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def get_crawler(self, name):  
        """  
        Gets information about a crawler.  
  
        :param name: The name of the crawler to look up.  
        :return: Data about the crawler.  
        """  
        crawler = None  
        try:  
            response = self.glue_client.get_crawler(Name=name)  
            crawler = response['Crawler']  
        except ClientError as err:  
            if err.response['Error']['Code'] == 'EntityNotFoundException':  
                logger.info("Crawler %s doesn't exist.", name)  
            else:  
                logger.error(  
                    "Couldn't get crawler %s. Here's why: %s: %s", name,  
                    err.response['Error']['Code'], err.response['Error'][  
                        'Message'])  
                raise  
        return crawler  
  
    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):  
        """  
        Creates a crawler that can crawl the specified target and populate a  
        database in your AWS Glue Data Catalog with metadata that describes the  
        data  
        in the target.  
  
        :param name: The name of the crawler.  
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and  
        Access  
            Management (IAM) role that grants permission to let AWS  
        Glue  
            access the resources it needs.  
        :param db_name: The name to give the database that is created by the  
        crawler.  
        :param db_prefix: The prefix to give any database tables that are created  
        by  
            the crawler.  
        :param s3_target: The URL to an S3 bucket that contains data that is  
        the target of the crawler.  
        """  
        try:  
            self.glue_client.create_crawler(  
                Name=name,  
                Role=role_arn,  
                DatabaseName=db_name,  
                TablePrefix=db_prefix,  
                Targets={'S3Targets': [{'Path': s3_target}]})  
        except ClientError as err:  
            logger.error(  
                "Couldn't create crawler. Here's why: %s: %s",  
                err.response['Error']['Code'], err.response['Error'][  
                    'Message'])  
            raise  
  
    def start_crawler(self, name):
```

```
"""
Starts a crawler. The crawler crawls its configured target and creates
metadata that describes the data it finds in the target data source.

:param name: The name of the crawler to start.
"""
try:
    self.glue_client.start_crawler(Name=name)
except ClientError as err:
    logger.error(
        "Couldn't start crawler %s. Here's why: %s: %s",
        name,
        err.response['Error']['Code'],
        err.response['Error']['Message']
    )
    raise

def get_database(self, name):
    """
    Gets information about a database in your Data Catalog.

    :param name: The name of the database to look up.
    :return: Information about the database.
    """
    try:
        response = self.glue_client.get_database(Name=name)
    except ClientError as err:
        logger.error(
            "Couldn't get database %s. Here's why: %s: %s",
            name,
            err.response['Error']['Code'],
            err.response['Error']['Message']
        )
        raise
    else:
        return response['Database']

def get_tables(self, db_name):
    """
    Gets a list of tables in a Data Catalog database.

    :param db_name: The name of the database to query.
    :return: The list of tables in the database.
    """
    try:
        response = self.glue_client.get_tables(DatabaseName=db_name)
    except ClientError as err:
        logger.error(
            "Couldn't get tables %s. Here's why: %s: %s",
            db_name,
            err.response['Error']['Code'],
            err.response['Error']['Message']
        )
        raise
    else:
        return response['TableList']

def create_job(self, name, description, role_arn, script_location):
    """
    Creates a job definition for an extract, transform, and load (ETL) job that
    can be run by AWS Glue.

    :param name: The name of the job definition.
    :param description: The description of the job definition.
    :param role_arn: The ARN of an IAM role that grants AWS Glue the
    permissions
        it requires to run the job.
    :param script_location: The Amazon S3 URL of a Python ETL script that is
    run as
        part of the job. The script defines how the data is
    transformed.
    """
    try:
        self.glue_client.create_job(
```

```

        Name=name, Description=description, Role=role_arn,
        Command={'Name': 'glueetl', 'ScriptLocation': script_location,
'PythonVersion': '3'},
        GlueVersion='3.0')
    except ClientError as err:
        logger.error(
            "Couldn't create job %s. Here's why: %s: %s", name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

    def start_job_run(self, name, input_database, input_table, output_bucket_name):
        """
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
        :param input_database: The name of the metadata database that contains
        tables
                           that describe the source data. This is typically
        created
                           by a crawler.
        :param input_table: The name of the table in the metadata database that
                           describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
        try:
            # The custom Arguments that are passed to this function are used by the
            # Python ETL script to determine the location of input and output data.
            response = self.glue_client.start_job_run(
                JobName=name,
                Arguments={
                    '--input_database': input_database,
                    '--input_table': input_table,
                    '--output_bucket_url': f's3://{output_bucket_name}/'})
        except ClientError as err:
            logger.error(
                "Couldn't start job run %s. Here's why: %s: %s", name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['JobRunId']

    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['JobNames']

    def get_job_runs(self, job_name):
        """
        Gets information about runs that have been performed for a specific job
        definition.

        :param job_name: The name of the job definition to look up.
        :return: The list of job runs.
        """

```

```
"""
try:
    response = self.glue_client.get_job_runs(JobName=job_name)
except ClientError as err:
    logger.error(
        "Couldn't get job runs for %s. Here's why: %s: %s",
        job_name,
        err.response['Error']['Code'],
        err.response['Error']['Message'])
    raise
else:
    return response['JobRuns']

def get_job_run(self, name, run_id):
    """
    Gets information about a single job run.

    :param name: The name of the job definition for the run.
    :param run_id: The ID of the run.
    :return: Information about the run.
    """
    try:
        response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
    except ClientError as err:
        logger.error(
            "Couldn't get job run %s/%s. Here's why: %s: %s",
            name,
            run_id,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return response['JobRun']

def delete_job(self, job_name):
    """
    Deletes a job definition. This also deletes data about all runs that are
    associated with this job definition.

    :param job_name: The name of the job definition to delete.
    """
    try:
        self.glue_client.delete_job(JobName=job_name)
    except ClientError as err:
        logger.error(
            "Couldn't delete job %s. Here's why: %s: %s",
            job_name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise

def delete_table(self, db_name, table_name):
    """
    Deletes a table from a metadata database.

    :param db_name: The name of the database that contains the table.
    :param table_name: The name of the table to delete.
    """
    try:
        self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
    except ClientError as err:
        logger.error(
            "Couldn't delete table %s. Here's why: %s: %s",
            table_name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise

def delete_database(self, name):
    """
    Deletes a metadata database from your Data Catalog.

    :param name: The name of the database to delete.
    """
    try:
```

```

        self.glue_client.delete_database(Name=name)
    except ClientError as err:
        logger.error(
            "Couldn't delete database %s. Here's why: %s: %s",
            name,
            err.response['Error']['Code'],
            err.response['Error']['Message']
        )
        raise

    def delete_crawler(self, name):
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response['Error']['Code'],
                err.response['Error']['Message']
            )
            raise

```

Create a class that runs the scenario.

```

class GlueCrawlerJobScenario:
    """
    Encapsulates a scenario that shows how to create an AWS Glue crawler and job
    and use
    them to transform data from CSV to JSON format.
    """
    def __init__(self, glue_client, glue_service_role, glue_bucket):
        """
        :param glue_client: A Boto3 AWS Glue client.
        :param glue_service_role: An AWS Identity and Access Management (IAM) role
            that AWS Glue can assume to gain access to the
            resources it requires.
        :param glue_bucket: An S3 bucket that can hold a job script and output data
            from AWS Glue job runs.
        """
        self.glue_client = glue_client
        self.glue_service_role = glue_service_role
        self.glue_bucket = glue_bucket

    @staticmethod
    def wait(seconds, tick=12):
        """
        Waits for a specified number of seconds, while also displaying an animated
        spinner.

        :param seconds: The number of seconds to wait.
        :param tick: The number of frames per second used to animate the spinner.
        """
        progress = '|/-\\|'
        waited = 0
        while waited < seconds:
            for frame in range(tick):
                sys.stdout.write(f"\r{progress[frame % len(progress)]}")
                sys.stdout.flush()
                time.sleep(1/tick)
            waited += 1

    def upload_job_script(self, job_script):
        """
        Uploads a Python ETL script to an S3 bucket. The script is used by the AWS
        Glue

```

```
job to transform data.

:param job_script: The relative path to the job script.
"""
try:
    self.glue_bucket.upload_file(Filename=job_script, Key=job_script)
    print(f"Uploaded job script '{job_script}' to the example bucket.")
except S3UploadFailedError as err:
    logger.error("Couldn't upload job script. Here's why: %s", err)
    raise

def run(
        self, crawler_name, db_name, db_prefix, data_source, job_script,
        job_name):
"""
Runs the scenario. This is an interactive experience that runs at a command
prompt and asks you for input throughout.

:param crawler_name: The name of the crawler used in the scenario. If the
                     crawler does not exist, it is created.
:param db_name: The name to give the metadata database created by the
                crawler.
:param db_prefix: The prefix to give tables added to the database by the
                  crawler.
:param data_source: The location of the data source that is targeted by the
                    crawler and extracted during job runs.
:param job_script: The job script that is used to transform data during job
                   runs.
:param job_name: The name to give the job definition that is created during
                 the
                 scenario.
"""
wrapper = GlueWrapper(self.glue_client)
print(f"Checking for crawler {crawler_name}.")
crawler = wrapper.get_crawler(crawler_name)
if crawler is None:
    print(f"Creating crawler {crawler_name}.")
    wrapper.create_crawler(
        crawler_name, self.glue_service_role.arn, db_name, db_prefix,
        data_source)
    print(f"Created crawler {crawler_name}.")
    crawler = wrapper.get_crawler(crawler_name)
    pprint(crawler)
    print('*'*88)

    print(f"When you run the crawler, it crawls data stored in {data_source}
and "
          f"creates a metadata database in the AWS Glue Data Catalog that
describes "
          f"the data in the data source.")
    print("In this example, the source data is in CSV format.")
    ready = False
    while not ready:
        ready = Question.ask_question(
            "Ready to start the crawler? (y/n) ", Question.is_yesno)
        wrapper.start_crawler(crawler_name)
    print("Let's wait for the crawler to run. This typically takes a few
minutes.")
    crawler_state = None
    while crawler_state != 'READY':
        self.wait(10)
        crawler = wrapper.get_crawler(crawler_name)
        crawler_state = crawler['State']
        print(f"Crawler is {crawler['State']}")
    print('*'*88)
```

```

database = wrapper.get_database(db_name)
print(f"The crawler created database {db_name}:")
pprint(database)
print(f"The database contains these tables:")
tables = wrapper.get_tables(db_name)
for index, table in enumerate(tables):
    print(f"\t{index + 1}. {table['Name']}")

table_index = Question.ask_question(
    f"Enter the number of a table to see more detail: ",
    Question.is_int, Question.in_range(1, len(tables)))
pprint(tables[table_index - 1])
print('*'*88)

print(f"Creating job definition {job_name}.")
wrapper.create_job(
    job_name, "Getting started example job.", self.glue_service_role.arn,
    f's3://{self.glue_bucket.name}/{job_script}')
print("Created job definition.")
print(f"When you run the job, it extracts data from {data_source},
transforms it "
      f"by using the {job_script} script, and loads the output into "
      f"S3 bucket {self.glue_bucket.name}.")
print("In this example, the data is transformed from CSV to JSON, and only
a few "
      "fields are included in the output.")

job_run_status = None
if Question.ask_question(f"Ready to run? (y/n)", Question.is_yesno):
    job_run_id = wrapper.start_job_run(
        job_name, db_name, tables[0]['Name'], self.glue_bucket.name)
    print(f"Job {job_name} started. Let's wait for it to run.")
    while job_run_status not in ['SUCCEEDED', 'STOPPED', 'FAILED',
'TIMEOUT']:
        self.wait(10)
        job_run = wrapper.get_job_run(job_name, job_run_id)
        job_run_status = job_run['JobRunState']
        print(f"Job {job_name}/{job_run_id} is {job_run_status}.")
    print('*'*88)

    if job_run_status == 'SUCCEEDED':
        print(f"Data from your job run is stored in your S3 bucket
'{self.glue_bucket.name}':")
        try:
            keys = [obj.key for obj in
self.glue_bucket.objects.filter(Prefix='run-')]
            for index, key in enumerate(keys):
                print(f"\t{index + 1}: {key}")
            lines = 4
            key_index = Question.ask_question(
                f"Enter the number of a block to download it and see the first
{lines} "
                f"lines of JSON output in the block: ",
                Question.is_int, Question.in_range(1, len(keys)))
            job_data = io.BytesIO()
            self.glue_bucket.download_fileobj(keys[key_index - 1], job_data)
            job_data.seek(0)
            for _ in range(lines):
                print(job_data.readline().decode('utf-8'))
        except ClientError as err:
            logger.error(
                "Couldn't get job run data. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error'])
            raise
        print('*'*88)

job_names = wrapper.list_jobs()

```

```

        if job_names:
            print(f"Your account has {len(job_names)} jobs defined:")
            for index, job_name in enumerate(job_names):
                print(f"\t{index + 1}. {job_name}")
            job_index = Question.ask_question(
                f"Enter a number between 1 and {len(job_names)} to see the list of
runs for "
                f"a job: ", Question.is_int, Question.in_range(1, len(job_names)))
            job_runs = wrapper.get_job_runs(job_names[job_index - 1])
            if job_runs:
                print(f"Found {len(job_runs)} runs for job {job_names[job_index - 1]}:")
                for index, job_run in enumerate(job_runs):
                    print(
                        f"\t{index + 1}. {job_run['JobRunState']} on "
                        f"{job_run['CompletedOn']:%Y-%m-%d %H:%M:%S}")
                run_index = Question.ask_question(
                    f"Enter a number between 1 and {len(job_runs)} to see details
for a run: ",
                    Question.is_int, Question.in_range(1, len(job_runs)))
                pprint(job_runs[run_index - 1])
            else:
                print(f"No runs found for job {job_names[job_index - 1]}")
        else:
            print("Your account doesn't have any jobs defined.")
    print('*'*88)

    print(f"Let's clean up. During this example we created job definition
'{job_name}'.")
    if Question.ask_question(
        "Do you want to delete the definition and all runs? (y/n) ",
        Question.is_yesno):
        wrapper.delete_job(job_name)
        print(f"Job definition '{job_name}' deleted.")
    tables = wrapper.get_tables(db_name)
    print(f"We also created database '{db_name}' that contains these tables:")
    for table in tables:
        print(f"\t{table['Name']}")
    if Question.ask_question(
        "Do you want to delete the tables and the database? (y/n) ",
        Question.is_yesno):
        for table in tables:
            wrapper.delete_table(db_name, table['Name'])
            print(f"Deleted table {table['Name']}.")

        wrapper.delete_database(db_name)
        print(f"Deleted database {db_name}.")
    print(f"We also created crawler '{crawler_name}'.")
    if Question.ask_question(
        "Do you want to delete the crawler? (y/n) ", Question.is_yesno):
        wrapper.delete_crawler(crawler_name)
        print(f"Deleted crawler {crawler_name}.")
    print('*'*88)

def parse_args(args):
    """
    Parse command line arguments.

    :param args: The command line arguments.
    :return: The parsed arguments.
    """
    parser = argparse.ArgumentParser(
        description="Runs the AWS Glue getting started with crawlers and jobs
scenario. "
                    "Before you run this scenario, set up scaffold resources by
running "

```

```

        "'python scaffold.py deploy'.")
parser.add_argument(
    'role_name',
    help="The name of an IAM role that AWS Glue can assume. This role must
grant access "
        "to Amazon S3 and to the permissions granted by the AWSGlueServiceRole"
"
    "managed policy.")
parser.add_argument(
    'bucket_name',
    help="The name of an S3 bucket that AWS Glue can access to get the job
script and "
        "put job results.")
parser.add_argument(
    '--job_script',
    default='flight_etl_job_script.py',
    help="The name of the job script file that is used in the scenario.")
return parser.parse_args(args)

def main():
    args = parse_args(sys.argv[1:])
    try:
        print('-' * 88)
        print("Welcome to the AWS Glue getting started with crawlers and jobs
scenario.")
        print('-' * 88)
        scenario = GlueCrawlerJobScenario(
            boto3.client('glue'),
            boto3.resource('iam').Role(args.role_name),
            boto3.resource('s3').Bucket(args.bucket_name))
        scenario.upload_job_script(args.job_script)
        scenario.run(
            'doc-example-crawler', 'doc-example-database', 'doc-example-',
            's3://crawler-public-us-east-1/flight/2016/csv',
            args.job_script, 'doc-example-job')
        print('-' * 88)
        print("To destroy scaffold resources, including the IAM role and S3 bucket
"
            "used in this scenario, run 'python scaffold.py destroy'.")
        print("\nThanks for watching!")
        print('-' * 88)
    except Exception:
        logging.exception("Something went wrong with the example.")

```

Create an ETL script that is used by AWS Glue to extract, transform, and load data during job runs.

```

import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
--input_database      The name of a metadata database that is contained in your
                        AWS Glue Data Catalog and that contains tables that
describe
                        the data to be processed.
--input_table          The name of a table in the database that describes the data
to

```

```

        be processed.
--output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(sys.argv, [
    "JOB_NAME", "input_database", "input_table", "output_bucket_url"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args['input_database'],
    table_name=args['input_table'],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args['output_bucket_url'], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)
job.commit()

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateCrawler](#)
  - [CreateJob](#)
  - [DeleteCrawler](#)
  - [DeleteDatabase](#)
  - [DeleteJob](#)
  - [DeleteTable](#)

- [GetCrawler](#)
- [GetDatabase](#)
- [GetDatabases](#)
- [GetJob](#)
- [GetJobRun](#)
- [GetJobRuns](#)
- [GetTables](#)
- [ListJobs](#)
- [StartCrawler](#)
- [StartJobRun](#)

## Code examples for IAM using AWS SDKs

The following code examples show how to use AWS Identity and Access Management with an AWS software development kit (SDK).

### Code examples

- [Actions for IAM using AWS SDKs \(p. 863\)](#)
  - [Attach an IAM policy to a role using an AWS SDK \(p. 864\)](#)
  - [Attach an IAM policy to a user using an AWS SDK \(p. 873\)](#)
  - [Attach an inline policy to an IAM role using an AWS SDK \(p. 875\)](#)
  - [Create an IAM policy using an AWS SDK \(p. 876\)](#)
  - [Create an IAM policy version using an AWS SDK \(p. 884\)](#)
  - [Create an IAM role using an AWS SDK \(p. 885\)](#)
  - [Create an IAM service-linked role using an AWS SDK \(p. 892\)](#)
  - [Create an IAM user using an AWS SDK \(p. 896\)](#)
  - [Create an IAM access key using an AWS SDK \(p. 902\)](#)
  - [Create an alias for an IAM account using an AWS SDK \(p. 908\)](#)
  - [Create an inline IAM policy for a user using an AWS SDK \(p. 913\)](#)
  - [Delete an IAM policy using an AWS SDK \(p. 913\)](#)
  - [Delete an IAM role using an AWS SDK \(p. 917\)](#)
  - [Delete an IAM role policy using an AWS SDK \(p. 919\)](#)
  - [Delete an IAM server certificate using an AWS SDK \(p. 920\)](#)
  - [Delete an IAM service-linked role using an AWS SDK \(p. 922\)](#)
  - [Delete an IAM user using an AWS SDK \(p. 923\)](#)
  - [Delete an IAM access key using an AWS SDK \(p. 929\)](#)
  - [Delete an IAM account alias using an AWS SDK \(p. 936\)](#)
  - [Delete an inline IAM policy from a user using an AWS SDK \(p. 939\)](#)
  - [Detach an IAM policy from a role using an AWS SDK \(p. 940\)](#)
  - [Detach an IAM policy from a user using an AWS SDK \(p. 946\)](#)
  - [Generate a credential report from IAM using an AWS SDK \(p. 948\)](#)
  - [Get a credential report from IAM using an AWS SDK \(p. 949\)](#)
  - [Get a detailed IAM authorization report for your account using an AWS SDK \(p. 949\)](#)
  - [Get an IAM policy using an AWS SDK \(p. 950\)](#)
  - [Get an IAM policy version using an AWS SDK \(p. 955\)](#)
  - [Get an IAM role using an AWS SDK \(p. 956\)](#)

- Get an IAM server certificate using an AWS SDK (p. 960)
- Get a summary of account usage from IAM using an AWS SDK (p. 961)
- Get data about the last use of an IAM access key using an AWS SDK (p. 962)
- Get the IAM account password policy using an AWS SDK (p. 966)
- List SAML providers for IAM using an AWS SDK (p. 969)
- List a user's IAM access keys using an AWS SDK (p. 972)
- List IAM account aliases using an AWS SDK (p. 978)
- List IAM groups using an AWS SDK (p. 983)
- List inline policies for an IAM role using an AWS SDK (p. 987)
- List IAM policies using an AWS SDK (p. 991)
- List policies attached to an IAM role using an AWS SDK (p. 997)
- List IAM roles using an AWS SDK (p. 1002)
- List IAM server certificates using an AWS SDK (p. 1006)
- List IAM users using an AWS SDK (p. 1008)
- Update an IAM server certificate using an AWS SDK (p. 1016)
- Update an IAM user using an AWS SDK (p. 1018)
- Update an IAM access key using an AWS SDK (p. 1022)
- Scenarios for IAM using AWS SDKs (p. 1027)
  - Create an IAM user and assume a role with AWS STS using an AWS SDK (p. 1027)
  - Create read-only and read-write IAM users using an AWS SDK (p. 1078)
  - Manage IAM access keys using an AWS SDK (p. 1084)
  - Manage IAM policies using an AWS SDK (p. 1087)
  - Manage IAM roles using an AWS SDK (p. 1090)
  - Manage your IAM account using an AWS SDK (p. 1093)
  - Roll back an IAM policy version using an AWS SDK (p. 1097)
- Cross-service examples for IAM using AWS SDKs (p. 1098)
  - Create a long-lived Amazon EMR cluster and run several steps using an AWS SDK (p. 1098)
  - Create a short-lived Amazon EMR cluster and run a step using an AWS SDK (p. 1099)

## Actions for IAM using AWS SDKs

The following code examples show how to use AWS Identity and Access Management with AWS SDKs. Each example calls an individual service function.

### Examples

- Attach an IAM policy to a role using an AWS SDK (p. 864)
- Attach an IAM policy to a user using an AWS SDK (p. 873)
- Attach an inline policy to an IAM role using an AWS SDK (p. 875)
- Create an IAM policy using an AWS SDK (p. 876)
- Create an IAM policy version using an AWS SDK (p. 884)
- Create an IAM role using an AWS SDK (p. 885)
- Create an IAM service-linked role using an AWS SDK (p. 892)
- Create an IAM user using an AWS SDK (p. 896)
- Create an IAM access key using an AWS SDK (p. 902)
- Create an alias for an IAM account using an AWS SDK (p. 908)
- Create an inline IAM policy for a user using an AWS SDK (p. 913)

- [Delete an IAM policy using an AWS SDK \(p. 913\)](#)
- [Delete an IAM role using an AWS SDK \(p. 917\)](#)
- [Delete an IAM role policy using an AWS SDK \(p. 919\)](#)
- [Delete an IAM server certificate using an AWS SDK \(p. 920\)](#)
- [Delete an IAM service-linked role using an AWS SDK \(p. 922\)](#)
- [Delete an IAM user using an AWS SDK \(p. 923\)](#)
- [Delete an IAM access key using an AWS SDK \(p. 929\)](#)
- [Delete an IAM account alias using an AWS SDK \(p. 936\)](#)
- [Delete an inline IAM policy from a user using an AWS SDK \(p. 939\)](#)
- [Detach an IAM policy from a role using an AWS SDK \(p. 940\)](#)
- [Detach an IAM policy from a user using an AWS SDK \(p. 946\)](#)
- [Generate a credential report from IAM using an AWS SDK \(p. 948\)](#)
- [Get a credential report from IAM using an AWS SDK \(p. 949\)](#)
- [Get a detailed IAM authorization report for your account using an AWS SDK \(p. 949\)](#)
- [Get an IAM policy using an AWS SDK \(p. 950\)](#)
- [Get an IAM policy version using an AWS SDK \(p. 955\)](#)
- [Get an IAM role using an AWS SDK \(p. 956\)](#)
- [Get an IAM server certificate using an AWS SDK \(p. 960\)](#)
- [Get a summary of account usage from IAM using an AWS SDK \(p. 961\)](#)
- [Get data about the last use of an IAM access key using an AWS SDK \(p. 962\)](#)
- [Get the IAM account password policy using an AWS SDK \(p. 966\)](#)
- [List SAML providers for IAM using an AWS SDK \(p. 969\)](#)
- [List a user's IAM access keys using an AWS SDK \(p. 972\)](#)
- [List IAM account aliases using an AWS SDK \(p. 978\)](#)
- [List IAM groups using an AWS SDK \(p. 983\)](#)
- [List inline policies for an IAM role using an AWS SDK \(p. 987\)](#)
- [List IAM policies using an AWS SDK \(p. 991\)](#)
- [List policies attached to an IAM role using an AWS SDK \(p. 997\)](#)
- [List IAM roles using an AWS SDK \(p. 1002\)](#)
- [List IAM server certificates using an AWS SDK \(p. 1006\)](#)
- [List IAM users using an AWS SDK \(p. 1008\)](#)
- [Update an IAM server certificate using an AWS SDK \(p. 1016\)](#)
- [Update an IAM user using an AWS SDK \(p. 1018\)](#)
- [Update an IAM access key using an AWS SDK \(p. 1022\)](#)

## Attach an IAM policy to a role using an AWS SDK

The following code examples show how to attach an IAM policy to a role.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Attach the policy to the role so that the user can assume it.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="policyArn">The ARN of the policy to attach.</param>
/// <param name="roleName">The name of the role to attach the policy to.</param>
public static async Task AttachRoleAsync(
    AmazonIdentityManagementServiceClient client,
    string policyArn,
    string roleName)
{
    var request = new AttachRolePolicyRequest
    {
        PolicyArn = policyArn,
        RoleName = roleName,
    };

    var response = await client.AttachRolePolicyAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine("Successfully attached the policy to the role.");
    }
    else
    {
        Console.WriteLine("Could not attach the policy.");
    }
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::attachRolePolicy(const Aws::String &roleName,
                                    const Aws::String &policyArn,
                                    const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::ListAttachedRolePoliciesRequest list_request;
    list_request.SetRoleName(roleName);

    bool done = false;
    while (!done) {
        auto list_outcome = iam.ListAttachedRolePolicies(list_request);
        if (!list_outcome.IsSuccess()) {
            std::cerr << "Failed to list attached policies of role " <<
                roleName << ":" << list_outcome.GetError().GetMessage() <<
                std::endl;
            return false;
    }
```

```
const auto & policies = list_outcome.GetResult().GetAttachedPolicies();
if (std::any_of(policies.cbegin(), policies.cend(),
                [=](const Aws::IAM::Model::AttachedPolicy & policy) {
                    return policy.GetPolicyArn() == policyArn;
                })) {
    std::cout << "Policy " << policyArn <<
        " is already attached to role " << roleName << std::endl;
    return true;
}

done = !list_outcome.GetResult().GetIsTruncated();
list_request.SetMarker(list_outcome.GetResult().GetMarker());
}

Aws::IAM::Model::AttachRolePolicyRequest request;
request.SetRoleName(roleName);
request.SetPolicyArn(policyArn);

Aws::IAM::Model::AttachRolePolicyOutcome outcome =
iam.AttachRolePolicy(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to attach policy " << policyArn << " to role " <<
        roleName << ": " << outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully attached policy " << policyArn << " to role " <<
        roleName << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// AttachRolePolicy

_, err = service.AttachRolePolicy(context.Background(),
&iام.AttachRolePolicyInput{
    PolicyArn: aws.String(ExamplePolicyARN),
    RoleName:  aws.String(ExampleRoleName),
})

if err != nil {
    panic("Couldn't apply a policy to the role!")
}

fmt.Println("## Attached policy " + ExamplePolicyARN + " to role " +
ExampleRoleName)
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String policyArn ) {

    try {
        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
            .roleName(roleName)
            .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        // Ensure that the policy is not attached to this role
        String polArn = "";
        for (AttachedPolicy policy: attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn)==0) {
                System.out.println(roleName + " policy is already attached to
this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.attachRolePolicy(attachRequest);

        System.out.println("Successfully attached policy " + policyArn +
" to role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Attach the policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import {
  ListAttachedRolePoliciesCommand,
  AttachRolePolicyCommand,
} from "@aws-sdk/client-iam";

// Set the parameters.
const ROLENAMESPACE = "ROLE_NAME";
const paramsRoleList = { RoleName: ROLENAMESPACE }; //ROLE_NAME
export const params = {
  PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
  RoleName: ROLENAMESPACE,
};
export const run = async () => {
  try {
    const data = await iamClient.send(
      new ListAttachedRolePoliciesCommand(paramsRoleList)
    );
    const myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (_val, index) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        console.log(
          "AmazonDynamoDBFullAccess is already attached to this role."
        );
        process.exit();
      }
    });
    try {
      const data = await iamClient.send(new AttachRolePolicyCommand(params));
      console.log("Role attached successfully");
      return data;
    } catch (err) {
      console.log("Error", err);
    }
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [AttachRolePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
```

```
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var paramsRoleList = {
  RoleName: process.argv[2]
};

iam.listAttachedRolePolicies(paramsRoleList, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === 'AmazonDynamoDBFullAccess') {
        console.log("AmazonDynamoDBFullAccess is already attached to this role.")
        process.exit();
      }
    });
    var params = {
      PolicyArn: 'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess',
      RoleName: process.argv[2]
    };
    iam.attachRolePolicy(params, function(err, data) {
      if (err) {
        console.log("Unable to attach policy to role", err);
      } else {
        console.log("Role attached successfully");
      }
    });
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [AttachRolePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun attachIAMRolePolicy(roleNameVal: String, policyArnVal: String) {

    val request = ListAttachedRolePoliciesRequest {
        roleName = roleNameVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies
```

```
// Ensure that the policy is not attached to this role.
val checkStatus: Int
    if (attachedPolicies != null) {
        checkStatus = checkList(attachedPolicies, policyArnVal)
        if (checkStatus == -1)
            return
    }

    val policyRequest = AttachRolePolicyRequest {
        roleName = roleNameVal
        policyArn = policyArnVal
    }
    iamClient.attachRolePolicy(policyRequest)
    println("Successfully attached policy $policyArnVal to role $roleNameVal")
}

fun checkList(attachedPolicies: List<AttachedPolicy>, policyArnVal: String): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

$assumeRolePolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Principal\": \"{\\\"AWS\\\": \"${user['Arn']}\"}\",
            \"Action\": \"sts:AssumeRole\"
        }
    ]
}";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
    $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": \"s3>ListAllMyBuckets\",
            \"Resource\": \"arn:aws:s3:::*\"}
    ]
}";
```

```
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
    $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\\n";

$listAllBucketsPolicy = $service->attachRolePolicy($assumeRoleRole['RoleName'],
    $listAllBucketsPolicy['Arn']);

    public function attachRolePolicy($roleName, $policyArn)
{
    return $this->customWaiter(function () use ($roleName, $policyArn) {
        $this->iامClient->attachRolePolicy([
            'PolicyArn' => $policyArn,
            'RoleName' => $roleName,
        ]);
    });
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Attach a policy to a role using the Boto3 Policy object.

```
def attach_to_role(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).attach_role(RoleName=role_name)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
                         role_name)
        raise
```

Attach a policy to a role using the Boto3 Role object.

```
def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
                         role_name)
        raise
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates a policy that grants permission to list S3 buckets in the account, and
# then attaches the policy to a role.
#
# @param policy_name [String] The name to give the policy.
# @param role [Aws::IAM::Role] The role that the policy is attached to.
# @return [Aws::IAM::Policy] The newly created policy.
def create_and_attach_role_policy(policy_name, role)
    policy = @iam_resource.create_policy(
        policy_name: policy_name,
        policy_document: {
            Version: "2012-10-17",
            Statement: [
                {
                    Effect: "Allow",
                    Action: "s3>ListAllMyBuckets",
                    Resource: "arn:aws:s3:::*"
                }
            ].to_json
        }
    )
    role.attach_policy(policy_arn: policy.arn)
    puts("Created policy #{policy.policy_name} and attached it to role
#{role.name}.")
    rescue Aws::Errors::ServiceError => e
        puts("Couldn't create a policy and attach it to role #{role.name}. Here's why:
")
        puts("\t#{e.code}: #{e.message}")
        raise
    else
        policy
    end
end
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn attach_role_policy(
    client: &iامClient,
```

```
        role: &Role,
        policy: &Policy,
    ) -> Result<AttachRolePolicyOutput, SdkError<AttachRolePolicyError>> {
    client
        .attach_role_policy()
        .role_name(role.role_name.as_ref().unwrap())
        .policy_arn(policy.arn.as_ref().unwrap())
        .send()
        .await
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Rust API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func attachRolePolicy(role: String, policyArn: String) async throws {
    let input = AttachRolePolicyInput(
        policyArn: policyArn,
        roleName: role
    )
    do {
        _ = try await client.attachRolePolicy(input: input)
    } catch {
        throw error
    }
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Swift API reference*.

## Attach an IAM policy to a user using an AWS SDK

The following code examples show how to attach an IAM policy to a user.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
```

```

"fmt"

"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMAttachRolePolicyAPI defines the interface for the AttachRolePolicy function.
// We use this interface to test the function using a mocked service.
type IAMAttachRolePolicyAPI interface {
    AttachRolePolicy(ctx context.Context,
        params *iam.AttachRolePolicyInput,
        optFns ...func(*iam.Options)) (*iam.AttachRolePolicyOutput, error)
}

// AttachDynamoFullPolicy attaches an Amazon DynamoDB full-access policy to an AWS
// Identity and Access Management (IAM) role.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, an AttachRolePolicyOutput object containing the result of the
//     service call and nil.
//     Otherwise, nil and an error from the call to AttachRolePolicy.
func AttachDynamoFullPolicy(c context.Context, api IAMAttachRolePolicyAPI, input
    *iam.AttachRolePolicyInput) (*iam.AttachRolePolicyOutput, error) {
    return api.AttachRolePolicy(c, input)
}

func main() {
    roleName := flag.String("r", "", "The name of the IAM role")
    policyName := flag.String("p", "", "The name of the policy to attach to the role")
    flag.Parse()

    if *roleName == "" || *policyName == "" {
        fmt.Println("You must supply a role and policy name (-r ROLE -p POLICY)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    policyArn := "arn:aws:iam::aws:policy/" + *policyName

    input := &iam.AttachRolePolicyInput{
        PolicyArn: &policyArn,
        RoleName:  roleName,
    }

    _, err = AttachDynamoFullPolicy(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Unable to attach policy " + *policyName + " to role " + *roleName)
        return
    }

    fmt.Println("Policy " + *policyName + " attached to role " + *roleName)
}

```

- For API details, see [AttachUserPolicy](#) in *AWS SDK for Go API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
                          user_name)
        raise
```

- For API details, see [AttachUserPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn attach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .attach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- For API details, see [AttachUserPolicy](#) in *AWS SDK for Rust API reference*.

## Attach an inline policy to an IAM role using an AWS SDK

The following code example shows how to attach an inline policy to an IAM role.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::putRolePolicy(
    const Aws::String &roleName,
    const Aws::String &policyName,
    const Aws::String &policyDocument,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iamClient(clientConfig);
    Aws::IAM::Model::PutRolePolicyRequest request;

    request.SetRoleName(roleName);
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(policyDocument);

    Aws::IAM::Model::PutRolePolicyOutcome outcome =
        iamClient.PutRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error putting policy on role. " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully put the role policy." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [PutRolePolicy](#) in *AWS SDK for C++ API Reference*.

## Create an IAM policy using an AWS SDK

The following code examples show how to create an IAM policy.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create a policy to allow a user to list the buckets in an account.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="policyName">The name of the policy to create.</param>
/// <param name="policyDocument">The permissions policy document.</param>
/// <returns>The newly created ManagedPolicy object.</returns>
public static async Task<ManagedPolicy> CreatePolicyAsync(
    AmazonIdentityManagementServiceClient client,
    string policyName,
    string policyDocument)
```

```
{  
    var request = new CreatePolicyRequest  
    {  
        PolicyName = policyName,  
        PolicyDocument = policyDocument,  
    };  
  
    var response = await client.CreatePolicyAsync(request);  
  
    return response.Policy;  
}
```

- For API details, see [CreatePolicy](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::String AwsDoc::IAM::createPolicy(const Aws::String &policyName,  
                                      const Aws::String &rsrcArn,  
                                      const Aws::Client::ClientConfiguration  
&clientConfig) {  
    Aws::IAM::IAMClient iam(clientConfig);  
  
    Aws::IAM::Model::CreatePolicyRequest request;  
    request.SetPolicyName(policyName);  
    request.SetPolicyDocument(BuildSamplePolicyDocument(rsrcArn));  
  
    Aws::IAM::Model::CreatePolicyOutcome outcome = iam.CreatePolicy(request);  
    Aws::String result;  
    if (!outcome.IsSuccess()) {  
        std::cerr << "Error creating policy " << policyName << ":" <<  
              outcome.GetError().GetMessage() << std::endl;  
    }  
    else {  
        result = outcome.GetResult().GetPolicy().GetArn();  
        std::cout << "Successfully created policy " << policyName <<  
              std::endl;  
    }  
  
    return result;  
}  
  
Aws::String AwsDoc::IAM::BuildSamplePolicyDocument(const Aws::String &rsrc_arn) {  
    std::stringstream stringStream;  
    stringStream << "{"  
        << "  \"Version\": \"2012-10-17\","  
        << "  \"Statement\": ["  
            << "    {"  
                << "      \"Effect\": \"Allow\","  
                << "      \"Action\": \"logs>CreateLogGroup\","  
                << "      \"Resource\": \"  
                << rsrc_arn  
                << "\""  
                << "    },"  
                << "    {"  
                    << "      \"Effect\": \"Allow\","
```

```
<< "      \\"Action\\": ["
<< "          \\\"dynamodb>DeleteItem\\\","
<< "          \\\"dynamodb>GetItem\\\","
<< "          \\\"dynamodb>PutItem\\\","
<< "          \\\"dynamodb>Scan\\\","
<< "          \\\"dynamodb>UpdateItem\\\""
<< "      ],"
<< "      \\\"Resource\\\": \\\""
<< rsrc_arn
<< "\\\""
<< "      }"
<< "  ]"
<< "};"

    return stringStream.str();
}
```

- For API details, see [CreatePolicy](#) in *AWS SDK for C++ API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// CreatePolicy

fmt.Println("# CreatePolicy")

policyDocument := `{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb>DeleteItem",
                "dynamodb>GetItem",
                "dynamodb>PutItem",
                "dynamodb>Query",
                "dynamodb>Scan",
                "dynamodb>UpdateItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/mytable",
                "arn:aws:dynamodb:us-west-2:123456789012:table/mytable/*"
            ]
        }
    ]
}`

createPolicyResult, err := service.CreatePolicy(context.Background(),
    &iam.CreatePolicyInput{
        PolicyDocument: &policyDocument,
        PolicyName:     aws.String(ExamplePolicyName),
    })
if err != nil {
    panic("Couldn't create policy!" + err.Error())
}
```

```
    fmt.Println("Created a new policy: " + *createPolicyResult.Policy.Arn)
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createIAMPolicy(IamClient iam, String policyName ) {  
  
    try {  
        // Create an IamWaiter object.  
        IamWaiter iamWaiter = iam.waiter();  
  
        CreatePolicyRequest request = CreatePolicyRequest.builder()  
            .policyName(policyName)  
            .policyDocument(PolicyDocument)  
            .build();  
  
        CreatePolicyResponse response = iam.createPolicy(request);  
  
        // Wait until the policy is created.  
        GetPolicyRequest polRequest = GetPolicyRequest.builder()  
            .policyArn(response.policy().arn())  
            .build();  
  
        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =  
            iamWaiter.waitUntilPolicyExists(polRequest);  
  
        waitUntilPolicyExists.matched().response().ifPresent(System.out::println);  
        return response.policy().arn();  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "" ;  
}
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";  
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".
```

```
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

Create the policy.

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "./libs/iamClient.js";  
import { CreatePolicyCommand } from "@aws-sdk/client-iam";  
  
// Set the parameters.  
const myManagedPolicy = {  
    Version: "2012-10-17",  
    Statement: [  
        {  
            Effect: "Allow",  
            Action: "logs:CreateLogGroup",  
            Resource: "RESOURCE_ARN", // RESOURCE_ARN  
        },  
        {  
            Effect: "Allow",  
            Action: [  
                "dynamodb>DeleteItem",  
                "dynamodb>GetItem",  
                "dynamodb>PutItem",  
                "dynamodb>Scan",  
                "dynamodb>UpdateItem",  
            ],  
            Resource: "DYNAMODB_POLICY_NAME", // DYNAMODB_POLICY_NAME; For example,  
            "myDynamoDBName".  
        },  
    ],  
};  
export const params = {  
    PolicyDocument: JSON.stringify(myManagedPolicy),  
    PolicyName: "IAM_POLICY_NAME",  
};  
  
export const run = async () => {  
    try {  
        const data = await iamClient.send(new CreatePolicyCommand(params));  
        console.log("Success", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreatePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');
```

```
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var myManagedPolicy = {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "RESOURCE_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb>DeleteItem",
                "dynamodb>GetItem",
                "dynamodb>PutItem",
                "dynamodb>Scan",
                "dynamodb>UpdateItem"
            ],
            "Resource": "RESOURCE_ARN"
        }
    ]
};

var params = {
    PolicyDocument: JSON.stringify(myManagedPolicy),
    PolicyName: 'myDynamoDBPolicy',
};

iam.createPolicy(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreatePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createIAMPolicy(policyNameVal: String?): String {

    val policyDocumentVal = "{" +
        "\"Version\": \"2012-10-17\", " +
        "\"Statement\": [ " +
```

```

        "    {" +
        "      \"Effect\": \"Allow\", " +
        "      \"Action\": [ " +
        "        \"dynamodb>DeleteItem\", " +
        "        \"dynamodb>GetItem\", " +
        "        \"dynamodb>PutItem\", " +
        "        \"dynamodb>Scan\", " +
        "        \"dynamodb>UpdateItem\" " +
        "      ], " +
        "      \"Resource\": \"*\\" +
        "    }" +
        "  ]" +
      "}" +
    "}" +
  "}" +
}

val request = CreatePolicyRequest {
  policyName = policyNameVal
  policyDocument = policyDocumentVal
}

IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
  val response = iamClient.createPolicy(request)
  return response.policy?.arn.toString()
}
}
}

```

- For API details, see [CreatePolicy](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

$uuid = uniqid();
 getService();

$listAllBucketsPolicyDocument = "{
  \"Version\": \"2012-10-17\",
  \"Statement\": [
    {
      \"Effect\": \"Allow\",
      \"Action\": \"s3>ListAllMyBuckets\",
      \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
  $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

  public function createPolicy(string $policyName, string $policyDocument)
  {
    $result = $this->customWaiter(function () use ($policyName,
$policyDocument) {
      return $this->iamClient->createPolicy([
        'PolicyName' => $policyName,
        'PolicyDocument' => $policyDocument,
      ]);
    });
    return $result['Policy'];
  }
}

```

- For API details, see [CreatePolicy](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_policy(name, description, actions, resource_arn):  
    """  
    Creates a policy that contains a single statement.  
  
    :param name: The name of the policy to create.  
    :param description: The description of the policy.  
    :param actions: The actions allowed by the policy. These typically take the  
        form of service:action, such as s3:PutObject.  
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy  
        applies to. This ARN can contain wildcards, such as  
        'arn:aws:s3:::my-bucket/*' to allow actions on all objects  
        in the bucket named 'my-bucket'.  
    :return: The newly created policy.  
    """  
    policy_doc = {  
        "Version": "2012-10-17",  
        "Statement": [  
            {  
                "Effect": "Allow",  
                "Action": actions,  
                "Resource": resource_arn  
            }  
        ]  
    }  
    try:  
        policy = iam.create_policy(  
            PolicyName=name, Description=description,  
            PolicyDocument=json.dumps(policy_doc))  
        logger.info("Created policy %s.", policy.arn)  
    except ClientError:  
        logger.exception("Couldn't create policy %s.", name)  
        raise  
    else:  
        return policy
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates a policy that grants permission to list S3 buckets in the account, and  
# then attaches the policy to a role.  
#  
# @param policy_name [String] The name to give the policy.
```

```
# @param role [Aws::IAM::Role] The role that the policy is attached to.  
# @return [Aws::IAM::Policy] The newly created policy.  
def create_and_attach_role_policy(policy_name, role)  
  policy = @iam_resource.create_policy(  
    policy_name: policy_name,  
    policy_document: {  
      Version: "2012-10-17",  
      Statement: [{  
        Effect: "Allow",  
        Action: "s3:listAllMyBuckets",  
        Resource: "arn:aws:s3:::*"  
      }]  
    }.to_json)  
  role.attach_policy(policy_arn: policy.arn)  
  puts("Created policy #{policy.policy_name} and attached it to role  
#{role.name}.")  
  rescue Aws::Errors::ServiceError => e  
    puts("Couldn't create a policy and attach it to role #{role.name}. Here's why:  
")  
    puts("\t#{e.code}: #{e.message}")  
    raise  
  else  
    policy  
  end
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn create_policy(  
    client: &iamClient,  
    policy_name: &str,  
    policy_document: &str,  
) -> Result<Policy, iamError> {  
    let policy = client  
        .create_policy()  
        .policy_name(policy_name)  
        .policy_document(policy_document)  
        .send()  
        .await?  
    Ok(policy.policy.unwrap())  
}
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Rust API reference*.

## Create an IAM policy version using an AWS SDK

The following code example shows how to create an IAM policy version.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                          version for the policy. Otherwise, the default
                          is not changed.
    :return: The newly created policy version.
    """

    policy_doc = {
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Action': actions,
                'Resource': resource_arn
            }
        ]
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default)
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id, policy_version.arn)
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version
```

- For API details, see [CreatePolicyVersion](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an IAM role using an AWS SDK

The following code examples show how to create an IAM role.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
```

```
    ///> Create a new IAM role which we can attach to a user.  
    ///> </summary>  
    ///> <param name="client">The initialized IAM client object.</param>  
    ///> <param name="roleName">The name of the IAM role to create.</param>  
    ///> <param name="rolePermissions">The permissions which the role will  
have.</param>  
    ///> <returns>A Role object representing the newly created role.</returns>  
    public static async Task<Role> CreateRoleAsync(  
        AmazonIdentityManagementServiceClient client,  
        string roleName,  
        string rolePermissions)  
    {  
        var request = new CreateRoleRequest  
        {  
            RoleName = roleName,  
            AssumeRolePolicyDocument = rolePermissions,  
        };  
  
        var response = await client.CreateRoleAsync(request);  
  
        return response.Role;  
    }
```

- For API details, see [CreateRole](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::createIamRole(  
    const Aws::String &roleName,  
    const Aws::String &policy,  
    const Aws::Client::ClientConfiguration &clientConfig) {  
    Aws::IAM::IAMClient client(clientConfig);  
    Aws::IAM::Model::CreateRoleRequest request;  
  
    request.SetRoleName(roleName);  
    request.SetAssumeRolePolicyDocument(policy);  
  
    Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);  
    if (!outcome.IsSuccess()) {  
        std::cerr << "Error creating role. " <<  
            outcome.GetError().GetMessage() << std::endl;  
    }  
    else {  
        const Aws::IAM::Model::Role iamRole = outcome.GetResult().GetRole();  
        std::cout << "Created role " << iamRole.GetRoleName() << "\n";  
        std::cout << "ID: " << iamRole.GetRoleId() << "\n";  
        std::cout << "ARN: " << iamRole.GetArn() << std::endl;  
    }  
  
    return outcome.IsSuccess();  
}
```

- For API details, see [CreateRole](#) in *AWS SDK for C++ API Reference*.

Go

**SDK for Go V2**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// CreateRole
myRole, err := service.CreateRole(context.Background(), &iam.CreateRoleInput{
    RoleName: aws.String(ExampleRoleName),
    Description: aws.String("My super awesome example role"),
    AssumeRolePolicyDocument: aws.String(`{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "ec2.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }`),
})
if err != nil {
    panic("Couldn't create role: " + err.Error())
}

fmt.Println("## Create Role")
fmt.Printf("The new role's ARN is %s \n", *myRole.Role.Arn)
```

- For API details, see [CreateRole](#) in *AWS SDK for Go API Reference*.

Java

**SDK for Java 2.x**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createIAMRole(IamClient iam, String rolename, String fileLocation ) throws Exception {

    try {
        JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(jsonObject.toJSONString())
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is "+response.role().arn());

    } catch (IamException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static Object readJsonSimpleDemo(String filename) throws Exception {
    FileReader reader = new FileReader(filename);
    JSONParser jsonParser = new JSONParser();
    return jsonParser.parse(reader);
}
```

- For API details, see [CreateRole](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Create the role.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { CreateRoleCommand } from "@aws-sdk/client-iam";

// Sample assume role policy JSON.
const role_json = {
    Version: "2012-10-17",
    Statement: [
        {
            Effect: "Allow",
            Principal: {
                AWS: "USER_ARN", // The ARN of the user.
            },
            Action: "sts:AssumeRole",
        },
    ],
};
// Stringify the assume role policy JSON.
const myJson = JSON.stringify(role_json);

// Set the parameters.
const params = {
    AssumeRolePolicyDocument: myJson,
    Path: "/",
    RoleName: "ROLE_NAME"
};
```

```
const run = async () => {
  try {
    const data = await iamClient.send(new CreateRoleCommand(params));
    console.log("Success. Role created. Role Arn: ", data.Role.RoleName);
  } catch (err) {
    console.log("Error", err);
  }
};

run();
```

- For API details, see [CreateRole](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
 getService();

$assumeRolePolicyDocument = "{
  \"Version\": \"2012-10-17\",
  \"Statement\": [
    {
      \"Effect\": \"Allow\",
      \"Principal\": \"{\\\"AWS\\\": \\"${user['Arn']}\\\"}\",
      \"Action\": \"sts:AssumeRole\"
    }
  ]
}";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
  $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

/**
 * @param string $roleName
 * @param string $rolePolicyDocument
 * @return array
 * @throws AwsException
 */
public function createRole(string $roleName, string $rolePolicyDocument)
{
    $result = $this->customWaiter(function () use ($roleName,
$rolePolicyDocument) {
        return $this->iamClient->createRole([
            'AssumeRolePolicyDocument' => $rolePolicyDocument,
            'RoleName' => $roleName,
        ]);
    });
    return $result['Role'];
}
```

- For API details, see [CreateRole](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Principal': {'Service': service},
                'Action': 'sts:AssumeRole'
            } for service in allowed_services
        ]
    }

    try:
        role = iam.create_role(
            RoleName=role_name,
            AssumeRolePolicyDocument=json.dumps(trust_policy))
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role
```

- For API details, see [CreateRole](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates a role that can be assumed by a user.
#
# @param role_name [String] The name to give the role.
# @param user [Aws::IAM::User] The user who is granted permission to assume the
# role.
# @return [Aws::IAM::Role] The newly created role.
def create_role(role_name, user)
  role = @iam_resource.create_role(
    role_name: role_name,
    assume_role_policy_document: {
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
```

```
        Principal: {'AWS': user.arn},
        Action: "sts:AssumeRole"
    }]
}.to_json)
puts("Created role #{role.name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't create a role for the demo. Here's why: ")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  role
end
```

- For API details, see [CreateRole](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn create_role(
    client: &iamClient,
    role_name: &str,
    role_policy_document: &str,
) -> Result<Role, iamError> {
    let response: CreateRoleOutput = loop {
        if let Ok(response) = client
            .create_role()
            .role_name(role_name)
            .assume_role_policy_document(role_policy_document)
            .send()
            .await
        {
            break response;
        }
    };
    Ok(response.role.unwrap())
}
```

- For API details, see [CreateRole](#) in *AWS SDK for Rust API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public func createRole(name: String, policyDocument: String) async throws ->
String {
    let input = CreateRoleInput(
        assumeRolePolicyDocument: policyDocument,
        roleName: name
    )
    do {
        let output = try await client.createRole(input: input)
        guard let role = output.role else {
            throw ServiceHandlerError.noSuchRole
        }
        guard let id = role.roleId else {
            throw ServiceHandlerError.noSuchRole
        }
        return id
    } catch {
        throw error
    }
}
```

- For API details, see [CreateRole](#) in *AWS SDK for Swift API reference*.

## Create an IAM service-linked role using an AWS SDK

The following code examples show how to create an IAM service-linked role.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// CreateServiceLinkedRole

fmt.Println("## Create SLR for " + ExampleSLRService)
createSlrResult, err := service.CreateServiceLinkedRole(context.Background(),
&iamp;.CreateServiceLinkedRoleInput{
    AWSServiceName: aws.String(ExampleSLRService),
    Description:   aws.String(ExampleSLRDescription),
})

// NOTE: We don't consider this an error as running this example multiple times
// will cause an error.
if err != nil {
    fmt.Printf("Couldn't create service-linked role: %v\n", err.Error())
} else {

    fmt.Printf("Created service-linked role with ARN: %s\n",
    *createSlrResult.Role.Arn)
}
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for Go API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Create a service-linked role.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { CreateServiceLinkedRoleCommand } from "@aws-sdk/client-iam";
// Set the parameters.
const params = {
  AWSServiceName: "AWS_SERVICE_NAME" /* required */,
};

const run = async () => {
  try {
    const data = await iamClient.send(
      new CreateServiceLinkedRoleCommand(params)
    );
    console.log("Success", data);
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

public function createServiceLinkedRole($awsServiceName, $customSuffix = "", $description = "") {
    $createServiceLinkedRoleArguments = ['AWSServiceName' => $awsServiceName];
    if ($customSuffix) {
        $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
    }
}
```

```
        if ($description) {
            $createServiceLinkedRoleArguments['Description'] = $description;
        }
        return $this->iamClient-
>createServiceLinkedRole($createServiceLinkedRoleArguments);
    }
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_service_linked_role(service_name, description):
    """
    Creates a service-linked role.

    :param service_name: The name of the service that owns the role.
    :param description: A description to give the role.
    :return: The newly created role.
    """
    try:
        response = iam.meta.client.create_service_linked_role(
            AWSServiceName=service_name, Description=description)
        role = iam.Role(response['Role']['RoleName'])
        logger.info("Created service-linked role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create service-linked role for %s.",
                         service_name)
        raise
    else:
        return role
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates a service-linked role.
#
# @param service_name [String] The name of the service that owns the role.
# @param description [String] A description to give the role.
# @return [Aws::IAM::Role] The newly created role.
def create_service_linked_role(service_name, description)
    response = @iam_resource.client.create_service_linked_role(
        aws_service_name: service_name, description: description)
    role = @iam_resource.role(response.role.role_name)
    puts("Created service-linked role #{role.name}.")
rescue Aws::Errors::ServiceError => e
```

```
    puts("Couldn't create service-linked role for #{service_name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
else
  role
end
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn create_service_linked_role(
    client: &iamClient,
    aws_service_name: String,
    custom_suffix: Option<String>,
    description: Option<String>,
) -> Result<CreateServiceLinkedRoleOutput, SdkError<CreateServiceLinkedRoleError>>
{
    let response = client
        .create_service_linked_role()
        .aws_service_name(aws_service_name)
        .set_custom_suffix(custom_suffix)
        .set_description(description)
        .send()
        .await?;

    Ok(response)
}
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for Rust API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func createServiceLinkedRole(service: String, suffix: String? = nil,
description: String?) {
    async throws -> IamClientTypes.Role {
    let input = CreateServiceLinkedRoleInput(
        aWSServiceName: service,
        customSuffix: suffix,
        description: description
```

```
)  
do {  
    let output = try await client.createServiceLinkedRole(input: input)  
    guard let role = output.role else {  
        throw ServiceHandlerError.noSuchRole  
    }  
    return role  
} catch {  
    throw error  
}  
}
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for Swift API reference*.

## Create an IAM user using an AWS SDK

The following code examples show how to create an IAM user.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>  
/// Create a new IAM user.  
///</summary>  
///<param name="client">The initialized IAM client object.</param>  
///<param name="userName">A string representing the user name of the  
/// new user.</param>  
///<returns>The newly created user.</returns>  
public static async Task<User> CreateUserAsync(  
    AmazonIdentityManagementServiceClient client,  
    string userName)  
{  
    var request = new CreateUserRequest  
    {  
        UserName = userName,  
    };  
  
    var response = await client.CreateUserAsync(request);  
  
    return response.User;  
}
```

- For API details, see [CreateUser](#) in *AWS SDK for .NET API Reference*.

### C++

#### SDK for C++

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::CreateUserRequest create_request;
create_request.SetUserName(userName);

auto create_outcome = iam.CreateUser(create_request);
if (!create_outcome.IsSuccess()) {
    std::cerr << "Error creating IAM user " << userName << ":" <<
        create_outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully created IAM user " << userName << std::endl;
}

return create_outcome.IsSuccess();
```

- For API details, see [CreateUser](#) in *AWS SDK for C++ API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// CreateUser

fmt.Println("## Create user " + ExampleUserName)

createUserResult, err := service.CreateUser(context.Background(),
&iam.CreateUserInput{
    UserName: aws.String(ExampleUserName),
})

if err != nil {
    panic("Couldn't create user: " + err.Error())
}

fmt.Printf("Created user %s\n", *createUserResult.User.Arn)
```

- For API details, see [CreateUser](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createIAMUser(IamClient iam, String username ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();
```

```
CreateUserRequest request = CreateUserRequest.builder()  
    .userName(username)  
    .build();  
  
CreateUserResponse response = iam.createUser(request);  
  
// Wait until the user is created  
GetUserRequest userRequest = GetUserRequest.builder()  
    .userName(response.user().userName())  
    .build();  
  
WaiterResponse<GetUserResponse> waitUntilUserExists =  
iamWaiter.waitUntilUserExists(userRequest);  
  
waitUntilUserExists.matched().response().ifPresent(System.out::println);  
return response.user().userName();  
  
} catch (IamException e) {  
    System.err.println(e.awsErrorDetails().errorMessage());  
    System.exit(1);  
}  
return "";  
}
```

- For API details, see [CreateUser](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";  
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

Create the user.

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "./libs/iamClient.js";  
import { GetUserCommand, CreateUserCommand } from "@aws-sdk/client-iam";  
  
// Set the parameters.  
export const params = { UserName: "USER_NAME" }; //USER_NAME  
  
export const run = async () => {  
    try {  
        const data = await iamClient.send(new GetUserCommand(params));  
        console.log(  
            "User " + "USER_NAME" + " already exists",  
            data.User.UserId  
        );  
        return data;  
    } catch (err) {
```

```
    try {
      const results = await iamClient.send(new CreateUserCommand(params));
      console.log("Success", results);
      return results;
    } catch (err) {
      console.log("Error", err);
    }
  };
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateUser](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  UserName: process.argv[2]
};

iam.getUser(params, function(err, data) {
  if (err && err.code === 'NoSuchEntity') {
    iam.createUser(params, function(err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  } else {
    console.log("User " + process.argv[2] + " already exists", data.UserId);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateUser](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createIAMUser(usernameVal: String?): String? {

    val request = CreateUserRequest {
        userName = usernameVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- For API details, see [CreateUser](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
 getService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

/**
 * @param string $name
 * @return array
 * @throws AwsException
 */
public function createUser(string $name): array
{
    $result = $this->iamClient->createUser([
        'UserName' => $name,
    ]);

    return $result['User'];
}
```

- For API details, see [CreateUser](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_user(user_name):
    """
```

```
Creates a user. By default, a user has no permissions or access keys.
```

```
:param user_name: The name of the user.  
:return: The newly created user.  
"""  
try:  
    user = iam.create_user(UserName=user_name)  
    logger.info("Created user %s.", user.name)  
except ClientError:  
    logger.exception("Couldn't create user %s.", user_name)  
    raise  
else:  
    return user
```

- For API details, see [CreateUser in AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates a user.  
#  
# @param user_name [String] The name to give the user.  
# @return [Aws::IAM::User] The newly created user.  
def create_user(user_name)  
    user = @iam_resource.create_user(user_name: user_name)  
    puts("Created demo user named #{user.name}.")  
rescue Aws::Errors::ServiceError => e  
    puts("Tried and failed to create demo user.")  
    puts("\t#{e.code}: #{e.message}")  
    puts("\nCan't continue the demo without a user!")  
    raise  
else  
    user  
end
```

- For API details, see [CreateUser in AWS SDK for Ruby API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn create_user(client: &iamClient, user_name: &str) -> Result<User, iamError> {  
    let response = client.create_user().user_name(user_name).send().await?;
```

```
        Ok(response.user.unwrap())
    }
```

- For API details, see [CreateUser](#) in *AWS SDK for Rust API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func createUser(name: String) async throws -> String {
    let input = CreateUserInput(
        userName: name
    )
    do {
        let output = try await client.createUser(input: input)
        guard let user = output.user else {
            throw ServiceHandlerError.noSuchUser
        }
        guard let id = user.userId else {
            throw ServiceHandlerError.noSuchUser
        }
        return id
    } catch {
        throw error
    }
}
```

- For API details, see [CreateUser](#) in *AWS SDK for Swift API reference*.

## Create an IAM access key using an AWS SDK

The following code examples show how to create an IAM access key.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create a new AccessKey for the user.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="userName">The name of the user for whom to create the
key.</param>
```

```
/// <returns>A new IAM access key for the user.</returns>
public static async Task<AccessKey> CreateAccessKeyAsync(
    AmazonIdentityManagementServiceClient client,
    string userName)
{
    var request = new CreateAccessKeyRequest
    {
        UserName = userName,
    };

    var response = await client.CreateAccessKeyAsync(request);

    if (response.AccessKey is not null)
    {
        Console.WriteLine($"Successfully created Access Key for
{userName}.");
    }

    return response.AccessKey;
}
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::String AwsDoc::IAM::createAccessKey(const Aws::String &userName,
                                         const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateAccessKeyRequest request;
    request.SetUserName(userName);

    Aws::String result;
    Aws::IAM::Model::CreateAccessKeyOutcome outcome = iam.CreateAccessKey(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating access key for IAM user " << userName
        << ":" << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &accessKey = outcome.GetResult().GetAccessKey();
        std::cout << "Successfully created access key for IAM user " <<
            userName << std::endl << " aws_access_key_id = " <<
            accessKey.GetAccessKeyId() << std::endl <<
            " aws_secret_access_key = " << accessKey.GetSecretAccessKey() <<
            std::endl;
        result = accessKey.GetAccessKeyId();
    }
}

return result;
}
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMCreateAccessKeyAPI defines the interface for the CreateAccessKey function.
// We use this interface to test the function using a mocked service.
type IAMCreateAccessKeyAPI interface {
    CreateAccessKey(ctx context.Context,
        params *iam.CreateAccessKeyInput,
        optFns ...func(*iam.Options)) (*iam.CreateAccessKeyOutput, error)
}

// MakeAccessKey creates a new AWS Identity and Access Management (IAM) access key
// for a user.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, a CreateAccessKeyOutput object containing the result of the
//     service call and nil.
//     Otherwise, nil and an error from the call to CreateAccessKey.
func MakeAccessKey(c context.Context, api IAMCreateAccessKeyAPI, input
    *iam.CreateAccessKeyInput) (*iam.CreateAccessKeyOutput, error) {
    return api.CreateAccessKey(c, input)
}

func main() {
    userName := flag.String("u", "", "The name of the user")
    flag.Parse()

    if *userName == "" {
        fmt.Println("You must supply a user name (-u USER)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.CreateAccessKeyInput{
        UserName: userName,
    }

    result, err := MakeAccessKey(context.TODO(), client, input)
    if err != nil {
```

```
    fmt.Println("Got an error creating a new access key")
    fmt.Println(err)
    return
}

fmt.Println("Created new access key with ID: " + *result.AccessKeyId + " "
and secret key: " + *result.AccessKey.SecretAccessKey)
}
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createIAMAccessKey(IamClient iam, String user) {

    try {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
            .userName(user)
            .build();

        CreateAccessKeyResponse response = iam.createAccessKey(request);
        return response.accessKey().accessKeyId();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Create the access key.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { CreateAccessKeyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {UserName: "IAM_USER_NAME"}; //IAM_USER_NAME

export const run = async () => {
  try {
    const data = await iamClient.send(new CreateAccessKeyCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateAccessKey](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.createAccessKey({UserName: 'IAM_USER_NAME'}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.AccessKey);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateAccessKey](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createIAMAccessKey(user: String?): String {
```

```
    val request = CreateAccessKeyRequest {
        userName = user
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """

    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name, key_pair.id)
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates an access key for a user.
#
# @param user [Aws::IAM::User] The user that owns the key.
# @return [Aws::IAM::AccessKeyPair] The newly created access key.
def create_access_key_pair(user)
    user_key = user.create_access_key_pair
    puts("Created access key pair for user.")
rescue Aws::Errors::ServiceError => e
```

```
    puts("Couldn't create access keys for user #{user.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
else
  user_key
end
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn create_access_key(client: &iamClient, user_name: &str) ->
Result<AccessKey, iamError> {
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<CreateAccessKeyOutput, SdkError<CreateAccessKeyError>> =
loop {
    match client.create_access_key().user_name(user_name).send().await {
        Ok(inner_response) => {
            break Ok(inner_response);
        }
        Err(e) => {
            tries += 1;
            if tries > max_tries {
                break Err(e);
            }
            sleep(Duration::from_secs(2)).await;
        }
    };
}

Ok(response.unwrap().access_key.unwrap())
}
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for Rust API reference*.

## Create an alias for an IAM account using an AWS SDK

The following code examples show how to create an alias for an IAM account.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::createAccountAlias(const Aws::String &aliasName,
                                      const Aws::Client::ClientConfiguration
                                      &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::CreateAccountAliasRequest request;
    request.SetAccountAlias(aliasName);

    Aws::IAM::Model::CreateAccountAliasOutcome outcome = iam.CreateAccountAlias(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating account alias " << aliasName << ":" 
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created account alias " << aliasName <<
                     std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [CreateAccountAlias](#) in *AWS SDK for C++ API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMCreateAccountAliasAPI defines the interface for the CreateAccountAlias
// function.
// We use this interface to test the function using a mocked service.
type IAMCreateAccountAliasAPI interface {
    CreateAccountAlias(ctx context.Context,
        params *iam.CreateAccountAliasInput,
        optFns ...func(*iam.Options)) (*iam.CreateAccountAliasOutput, error)
}

// MakeAccountAlias creates an alias for your AWS Identity and Access Management
// (IAM) account.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If successful, a CreateAccountAliasOutput object containing the result of
//   the service call and nil.
//   Otherwise, nil and an error from the call to CreateAccountAlias.

```

```
func MakeAccountAlias(c context.Context, api IAMCreateAccountAliasAPI, input *iam.CreateAccountAliasInput) (*iam.CreateAccountAliasOutput, error) {
    return api.CreateAccountAlias(c, input)
}

func main() {
    alias := flag.String("a", "", "The account alias")
    flag.Parse()

    if *alias == "" {
        fmt.Println("You must supply an account alias (-a ALIAS)")
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.CreateAccountAliasInput{
        AccountAlias: alias,
    }

    _, err = MakeAccountAlias(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error creating an account alias")
        fmt.Println(err)
        return
    }

    fmt.Printf("Created account alias " + *alias)
}
```

- For API details, see [CreateAccountAlias](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createIAMAccountAlias(IamClient iam, String alias) {

    try {
        CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()
            .accountAlias(alias)
            .build();

        iam.createAccountAlias(request);
        System.out.println("Successfully created account alias: " + alias);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateAccountAlias](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Create the account alias.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { CreateAccountAliasCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { AccountAlias: "ACCOUNT_ALIAS" }; //ACCOUNT_ALIAS

export const run = async () => {
  try {
    const data = await iamClient.send(new CreateAccountAliasCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateAccountAlias](#) in *AWS SDK for JavaScript API Reference*.

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.createAccountAlias({AccountAlias: process.argv[2]}, function(err, data) {
  if (err) {
```

```
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateAccountAlias](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createIAMAccountAlias(alias: String) {

    val request = CreateAccountAliasRequest {
        accountAlias = alias
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- For API details, see [CreateAccountAlias](#) in [AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
```

```
raise
```

- For API details, see [CreateAccountAlias](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an inline IAM policy for a user using an AWS SDK

The following code example shows how to create an inline IAM policy for a user.

Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates an inline policy for a user that lets the user assume a role.  
#  
# @param policy_name [String] The name to give the policy.  
# @param user [Aws::IAM::User] The user that owns the policy.  
# @param role [Aws::IAM::Role] The role that can be assumed.  
# @return [Aws::IAM::UserPolicy] The newly created policy.  
def create_user_policy(policy_name, user, role)  
    policy = user.create_policy(  
        policy_name: policy_name,  
        policy_document: {  
            Version: "2012-10-17",  
            Statement: [{  
                Effect: "Allow",  
                Action: "sts:AssumeRole",  
                Resource: role.arn  
            }]  
        }.to_json)  
    puts("Created an inline policy for #{user.name} that lets the user assume role  
#{role.name}.")  
    rescue Aws::Errors::ServiceError => e  
        puts("Couldn't create an inline policy for user #{user.name}. Here's why: ")  
        puts("\t#{e.code}: #{e.message}")  
        raise  
    else  
        policy  
    end
```

- For API details, see [PutUserPolicy](#) in *AWS SDK for Ruby API Reference*.

## Delete an IAM policy using an AWS SDK

The following code examples show how to delete an IAM policy.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::deletePolicy(const Aws::String &policyArn,
                                const Aws::Client::ClientConfiguration
                                &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeletePolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.DeletePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting policy with arn " << policyArn << ": "
              << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted policy with arn " << policyArn
              << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeletePolicy](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteIAMPolicy(IamClient iam, String policyARN) {

    try {
        DeletePolicyRequest request = DeletePolicyRequest.builder()
            .policyArn(policyARN)
            .build();

        iam.deletePolicy(request);
        System.out.println("Successfully deleted the policy");

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

- For API details, see [DeletePolicy](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Delete the policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { DeletePolicyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const params = { PolicyArn: "POLICY_ARN" };

const run = async () => {
    try {
        const data = await iamClient.send(new DeletePolicyCommand(params));
        console.log("Success. Policy deleted.", data);
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For API details, see [DeletePolicy](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {

    val request = DeletePolicyRequest {
        policyArn = policyARNVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- For API details, see [DeletePolicy](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

- For API details, see [DeletePolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a role. If the role has policies attached, they are detached and
# deleted before the role is deleted.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_role(role)
    role.attached_policies.each do |policy|
        name = policy.policy_name
        policy.detach_role(role_name: role.name)
        policy.delete
        puts("Deleted policy #{name}.")
    end
    name = role.name
    role.delete
    puts("Deleted role #{name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't detach policies and delete role #{role.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
end
```

- For API details, see [DeletePolicy](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_policy(client: &iamClient, policy: Policy) -> Result<(),  
    iamError> {  
    client  
        .delete_policy()  
        .policy_arn(policy.arn.unwrap())  
        .send()  
        .await?  
    Ok(())  
}
```

- For API details, see [DeletePolicy](#) in *AWS SDK for Rust API reference*.

## Delete an IAM role using an AWS SDK

The following code examples show how to delete an IAM role.

### JavaScript

#### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";  
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

Delete the role.

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "./libs/iamClient.js";  
import { DeleteRoleCommand } from "@aws-sdk/client-iam";  
  
// Set the parameters.  
const params = {  
    RoleName: "ROLE_NAME"  
}  
  
const run = async () => {  
    try {
```

```
        const data = await iamClient.send(new DeleteRoleCommand(params));
        console.log("Success. Role deleted.", data);
    } catch (err) {
        console.log("Error", err);
    }
};

run();
```

- For API details, see [DeleteRole](#) in *AWS SDK for JavaScript API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

- For API details, see [DeleteRole](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a role. If the role has policies attached, they are detached and
# deleted before the role is deleted.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_role(role)
    role.attached_policies.each do |policy|
        name = policy.policy_name
        policy.detach_role(role_name: role.name)
        policy.delete
        puts("Deleted policy #{name}.")
    end
    name = role.name
    role.delete
    puts("Deleted role #{name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't detach policies and delete role #{role.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
```

```
    raise
end
```

- For API details, see [DeleteRole](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_role(client: &iamClient, role: &Role) -> Result<(), iamError> {
    let role = role.clone();
    loop {
        match client
            .delete_role()
            .role_name(role.role_name.as_ref().unwrap())
            .send()
            .await
        {
            Ok(_) => {
                break;
            }
            Err(_) => {
                sleep(Duration::from_secs(2)).await;
            }
        }
    }
    Ok(())
}
```

- For API details, see [DeleteRole](#) in *AWS SDK for Rust API reference*.

## Delete an IAM role policy using an AWS SDK

The following code example shows how to delete an IAM role policy.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

public class DeleteRolePolicy
```

```
{
    /// <summary>
    /// Initializes the IAM client object and then calls DeleteRolePolicyAsync
    /// to delete the Policy attached to the Role.
    /// </summary>
    public static async Task Main()
    {
        var client = new AmazonIdentityManagementServiceClient();
        var response = await client.DeleteRolePolicyAsync(new
DeleteRolePolicyRequest
        {
            PolicyName = "ExamplePolicy",
            RoleName = "Test-Role",
        });

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("Policy successfully deleted.");
        }
        else
        {
            Console.WriteLine("Could not delete policy.");
        }
    }
}
```

- For API details, see [DeleteRolePolicy](#) in *AWS SDK for .NET API Reference*.

## Delete an IAM server certificate using an AWS SDK

The following code examples show how to delete an IAM server certificate.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::deleteServerCertificate(const Aws::String &certificateName,
                                            const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeleteServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    const auto outcome = iam.DeleteServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error deleting server certificate " << certificateName <<
                ":" << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                << "' not found." << std::endl;
        }
    }
}
```

```
        }
    else {
        std::cout << "Successfully deleted server certificate " << certificateName
        << std::endl;
    }

    return result;
}
```

- For API details, see [DeleteServerCertificate](#) in *AWS SDK for C++ API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Delete a server certificate.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { DeleteServerCertificateCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { ServerCertificateName: "CERTIFICATE_NAME" }; // CERTIFICATE_NAME

export const run = async () => {
    try {
        const data = await iamClient.send(
            new DeleteServerCertificateCommand(params)
        );
        console.log("Success", data);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteServerCertificate](#) in *AWS SDK for JavaScript API Reference*.

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.deleteServerCertificate({ServerCertificateName: 'CERTIFICATE_NAME'},
  function(err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteServerCertificate](#) in *AWS SDK for JavaScript API Reference*.

## Delete an IAM service-linked role using an AWS SDK

The following code examples show how to delete an IAM service-linked role.

Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a service-linked role from the account.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_service_linked_role(role)
  response = @iam_resource.client.delete_service_linked_role(role_name:
role.name)
  task_id = response.deletion_task_id
  while true
    response = @iam_resource.client.get_service_linked_role_deletion_status(
      deletion_task_id: task_id)
    status = response.status
    puts("Deletion of #{role.name} #{status}.")
    if %w[SUCCEEDED FAILED].include?(status)
      break
    else
      sleep(3)
    end
  end
rescue Aws::Errors::ServiceError => e
  # If AWS has not yet fully propagated the role, it deletes the role but
  # returns NoSuchEntity.
  if e.code != "NoSuchEntity"
    puts("Couldn't delete #{role.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- For API details, see [DeleteServiceLinkedRole](#) in *AWS SDK for Ruby API Reference*.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_service_linked_role(
    client: &iamClient,
    role_name: &str,
) -> Result<(), iamError> {
    client
        .delete_service_linked_role()
        .role_name(role_name)
        .send()
        .await?;

    Ok(())
}
```

- For API details, see [DeleteServiceLinkedRole](#) in *AWS SDK for Rust API reference*.

## Delete an IAM user using an AWS SDK

The following code examples show how to delete an IAM user.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Delete the user, and other resources created for this example.
///</summary>
///<param name="client">The initialized client object.</param>
///<param name="accessKeyId">The Id of the user's access key.</param>
///<param name="userName">The user name of the user to delete.</param>
///<param name="policyName">The name of the policy to delete.</param>
///<param name="policyArn">The Amazon Resource Name ARN of the Policy to
delete.</param>
///<param name="roleName">The name of the role that will be deleted.</param>
public static async Task DeleteResourcesAsync(
    AmazonIdentityManagementServiceClient client,
    string accessKeyId,
    string userName,
```

```
        string policyArn,
        string roleName)
    {
        var detachPolicyResponse = await client.DetachRolePolicyAsync(new
DetachRolePolicyRequest
{
    PolicyArn = policyArn,
    RoleName = roleName,
});

        var delPolicyResponse = await client.DeletePolicyAsync(new
DeletePolicyRequest
{
    PolicyArn = policyArn,
});

        var delRoleResponse = await client.DeleteRoleAsync(new
DeleteRoleRequest
{
    RoleName = roleName,
});

        var delAccessKey = await client.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
{
    AccessKeyId = accessKeyId,
    UserName = userName,
});

        var delUserResponse = await client.DeleteUserAsync(new
DeleteUserRequest
{
    UserName = userName,
});
    }
}
```

- For API details, see [DeleteUser in AWS SDK for .NET API Reference](#).

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::DeleteUserRequest request;
request.SetUserName(userName);
auto outcome = iam.DeleteUser(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error deleting IAM user " << userName << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully deleted IAM user " << userName << std::endl;
}

return outcome.IsSuccess();
```

- For API details, see [DeleteUser](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// DeleteUser

-, err = service.DeleteUser(context.Background(), &iam.DeleteUserInput{
    UserName: aws.String(ExampleUserName),
})

if err != nil {
    panic("Couldn't delete user: " + err.Error())
}
```

- For API details, see [DeleteUser](#) in *AWS SDK for Go API Reference*.

Java

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteIAMUser(IamClient iam, String userName) {

    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("Successfully deleted IAM user " + userName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteUser](#) in *AWS SDK for Java 2.x API Reference*.

JavaScript

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Delete the user.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { DeleteUserCommand, GetUserCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { UserName: "USER_NAME" }; //USER_NAME

export const run = async () => {
    try {
        const data = await iamClient.send(new GetUserCommand(params));
        return data;
    } catch (err) {
        console.log("Error", err);
    }
} catch (err) {
    console.log("User " + "USER_NAME" + " does not exist.");
}
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteUser in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
    UserName: process.argv[2]
};

iam.getUser(params, function(err, data) {
    if (err && err.code === 'NoSuchEntity') {
        console.log("User " + process.argv[2] + " does not exist.");
    } else {
```

```
iam.deleteUser(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteUser](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteIAMUser(userNameVal: String) {

    val request = DeleteUserRequest {
        userName = userNameVal
    }

    // To delete a user, ensure that the user's access keys are deleted first.
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- For API details, see [DeleteUser](#) in [AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
```

```
    logger.info("Deleted user %s.", user_name)
except ClientError:
    logger.exception("Couldn't delete user %s.", user_name)
    raise
```

- For API details, see [DeleteUser in AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a user. If the user has inline policies or access keys, they are
# deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
    user.policies.each do |policy|
        name = policy.name
        policy.delete
        puts("Deleted user policy #{name}.")
    end
    user.access_keys.each do |key|
        key.delete
        puts("Deleted access key for user #{user.name}.")
    end
    name = user.name
    user.delete
    puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
end
```

- For API details, see [DeleteUser in AWS SDK for Ruby API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_user(client: &iamClient, user: &User) -> Result<(),
SdkError<DeleteUserError>> {
    let user = user.clone();
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;
```

```
let response: Result<(), SdkError<DeleteUserError>> = loop {
    match client
        .delete_user()
        .user_name(user.user_name.as_ref().unwrap())
        .send()
        .await
    {
        Ok(_) => {
            break Ok(());
        }
        Err(e) => {
            tries += 1;
            if tries > max_tries {
                break Err(e);
            }
            sleep(Duration::from_secs(2)).await;
        }
    }
};  
response
}
```

- For API details, see [DeleteUser in AWS SDK for Rust API reference](#).

## Delete an IAM access key using an AWS SDK

The following code examples show how to delete an IAM access key.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Delete the user, and other resources created for this example.
///</summary>
///<param name="client">The initialized client object.</param>
///<param name="accessKeyId">The Id of the user's access key.</param>
///<param name="userName">The user name of the user to delete.</param>
///<param name="policyName">The name of the policy to delete.</param>
///<param name="policyArn">The Amazon Resource Name ARN of the Policy to
delete.</param>
///<param name="roleName">The name of the role that will be deleted.</param>
public static async Task DeleteResourcesAsync(
    AmazonIdentityManagementServiceClient client,
    string accessKeyId,
    string userName,
    string policyArn,
    string roleName)
{
    var detachPolicyResponse = await client.DetachRolePolicyAsync(new
DetachRolePolicyRequest
{
    PolicyArn = policyArn,
    RoleName = roleName,
```

```
});

var delPolicyResponse = await client.DeletePolicyAsync(new
DeletePolicyRequest
{
    PolicyArn = policyArn,
});

var delRoleResponse = await client.DeleteRoleAsync(new
DeleteRoleRequest
{
    RoleName = roleName,
});

var delAccessKey = await client.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
{
    AccessKeyId = accessKeyId,
    UserName = userName,
});

var delUserResponse = await client.DeleteUserAsync(new
DeleteUserRequest
{
    UserName = userName,
});

}
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::deleteAccessKey(const Aws::String &userName,
                                    const Aws::String &accessKeyID,
                                    const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);

    auto outcome = iam.DeleteAccessKey(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting access key " << accessKeyID << " from user "
              << userName << ":" << outcome.GetError().GetMessage() <<
              std::endl;
    }
    else {
        std::cout << "Successfully deleted access key " << accessKeyID
              << " for IAM user " << userName << std::endl;
    }
}
```

```
        return outcome.IsSuccess();
}
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for C++ API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMDeleteAccessKeyAPI defines the interface for the DeleteAccessKey function.
// We use this interface to test the function using a mocked service.
type IAMDeleteAccessKeyAPI interface {
    DeleteAccessKey(ctx context.Context,
        params *iam.DeleteAccessKeyInput,
        optFns ...func(*iam.Options)) (*iam.DeleteAccessKeyOutput, error)
}

// RemoveAccessKey deletes an AWS Identity and Access Management (IAM) access key.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, a DeleteAccessKeyOutput object containing the result of the
//     service call and nil.
//     Otherwise, nil and an error from the call to DeleteAccessKey.
func RemoveAccessKey(c context.Context, api IAMDeleteAccessKeyAPI, input
    *iam.DeleteAccessKeyInput) (*iam.DeleteAccessKeyOutput, error) {
    return api.DeleteAccessKey(c, input)
}

func main() {
    keyID := flag.String("k", "", "The ID of the access key")
    userName := flag.String("u", "", "The name of the user")
    flag.Parse()

    if *keyID == "" || *userName == "" {
        fmt.Println("You must supply the key ID and user name (-k KEY-ID -u USER-NAME")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)
```

```
input := &iam.DeleteAccessKeyInput{
    AccessKeyId: keyID,
    UserName:    userName,
}

_, err = RemoveAccessKey(context.TODO(), client, input)
if err != nil {
    fmt.Println("Error", err)
    return
}

fmt.Println("Deleted key with ID " + *keyID + " from user " + *userName)
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteKey(IamClient iam ,String username, String accessKey )
{

    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
                           " from user " + username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
```

```
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

Delete the access key.

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "./libs/iamClient.js";  
import { DeleteAccessKeyCommand } from "@aws-sdk/client-iam";  
  
// Set the parameters.  
export const params = {  
    AccessKeyId: "ACCESS_KEY_ID", // ACCESS_KEY_ID  
    UserName: "USER_NAME", // USER_NAME  
};  
  
export const run = async () => {  
    try {  
        const data = await iamClient.send(new DeleteAccessKeyCommand(params));  
        console.log("Success", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAccessKey](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create the IAM service object  
var iam = new AWS.IAM({apiVersion: '2010-05-08'});  
  
var params = {  
    AccessKeyId: 'ACCESS_KEY_ID',  
    UserName: 'USER_NAME'  
};  
  
iam.deleteAccessKey(params, function(err, data) {  
    if (err) {  
        console.log("Error", err);  
    } else {  
        console.log("Success", data);  
    }  
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAccessKey](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteKey(userNameVal: String, accessKey: String) {  
  
    val request = DeleteAccessKeyRequest {  
        accessKeyId = accessKey  
        userName = userNameVal  
    }  
  
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        iamClient.deleteAccessKey(request)  
        println("Successfully deleted access key $accessKey from $userNameVal")  
    }  
}
```

- For API details, see [DeleteAccessKey](#) in [AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_key(user_name, key_id):  
    """  
    Deletes a user's access key.  
  
    :param user_name: The user that owns the key.  
    :param key_id: The ID of the key to delete.  
    """  
  
    try:  
        key = iam.AccessKey(user_name, key_id)  
        key.delete()  
        logger.info(  
            "Deleted access key %s for %s.", key.id, key.user_name)  
    except ClientError:  
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)  
        raise
```

- For API details, see [DeleteAccessKey](#) in [AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a user. If the user has inline policies or access keys, they are deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
    user.policies.each do |policy|
        name = policy.name
        policy.delete
        puts("Deleted user policy #{name}.")
    end
    user.access_keys.each do |key|
        key.delete
        puts("Deleted access key for user #{user.name}.")
    end
    name = user.name
    user.delete
    puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
end
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_access_key(
    client: &iamClient,
    user: &User,
    key: &AccessKey,
) -> Result<(), iamError> {
    loop {
        match client
            .delete_access_key()
            .user_name(user.user_name.as_ref().unwrap())
            .access_key_id(key.access_key_id.as_ref().unwrap())
            .send()
            .await
        {
            Ok(_) => {
                break;
            }
        }
    }
}
```

```
        Err(e) => {
            println!("Can't delete the access key: {:?}", e);
            sleep(Duration::from_secs(2)).await;
        }
    }
Ok(())
}
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for Rust API reference*.

## Delete an IAM account alias using an AWS SDK

The following code examples show how to delete an IAM account alias.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::deleteAccountAlias(const Aws::String &accountAlias,
                                       const Aws::Client::ClientConfiguration
                                       &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccountAliasRequest request;
    request.SetAccountAlias(accountAlias);

    const auto outcome = iam.DeleteAccountAlias(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting account alias " << accountAlias << ": "
              << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted account alias " << accountAlias <<
                     std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteAccountAlias](#) in *AWS SDK for C++ API Reference*.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteIAMAccountAlias(IamClient iam, String alias ) {

    try {
        DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()
```

```
        .accountAlias(alias)
        .build();

    iam.deleteAccountAlias(request);
    System.out.println("Successfully deleted account alias " + alias);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.println("Done");
}
```

- For API details, see [DeleteAccountAlias](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Delete the account alias.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { DeleteAccountAliasCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { AccountAlias: "ALIAS" }; // ALIAS

export const run = async () => {
    try {
        const data = await iamClient.send(new DeleteAccountAliasCommand(params));
        console.log("Success", data);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAccountAlias](#) in *AWS SDK for JavaScript API Reference*.

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.deleteAccountAlias({AccountAlias: process.argv[2]}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAccountAlias](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteIAMAccountAlias(alias: String) {

    val request = DeleteAccountAliasRequest {
        accountAlias = alias
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- For API details, see [DeleteAccountAlias](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_alias(alias):
    """
```

```
Removes the alias from the current account.

:param alias: The alias to remove.
"""
try:
    iam.meta.client.delete_account_alias(AccountAlias=alias)
    logger.info("Removed alias '%s' from your account.", alias)
except ClientError:
    logger.exception("Couldn't remove alias '%s' from your account.", alias)
    raise
```

- For API details, see [DeleteAccountAlias](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an inline IAM policy from a user using an AWS SDK

The following code examples show how to delete an inline IAM policy from a user.

Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a user. If the user has inline policies or access keys, they are
# deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
  user.policies.each do |policy|
    name = policy.name
    policy.delete
    puts("Deleted user policy #{name}.")
  end
  user.access_keys.each do |key|
    key.delete
    puts("Deleted access key for user #{user.name}.")
  end
  name = user.name
  user.delete
  puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
end
```

- For API details, see [DeleteUserPolicy](#) in *AWS SDK for Ruby API Reference*.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_user_policy(
    client: &iamClient,
    user: &User,
    policy_name: &str,
) -> Result<(), SdkError<DeleteUserPolicyError>> {
    client
        .delete_user_policy()
        .user_name(user.user_name.as_ref().unwrap())
        .policy_name(policy_name)
        .send()
        .await?;

    Ok(())
}
```

- For API details, see [DeleteUserPolicy](#) in *AWS SDK for Rust API reference*.

## Detach an IAM policy from a role using an AWS SDK

The following code examples show how to detach an IAM policy from a role.

.NET

### AWS SDK for .NET

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Delete the user, and other resources created for this example.
///</summary>
///<param name="client">The initialized client object.</param>
///<param name="accessKeyId">The Id of the user's access key.</param>
///<param name="userName">The user name of the user to delete.</param>
///<param name="policyName">The name of the policy to delete.</param>
///<param name="policyArn">The Amazon Resource Name ARN of the Policy to
delete.</param>
///<param name="roleName">The name of the role that will be deleted.</param>
public static async Task DeleteResourcesAsync(
    AmazonIdentityManagementServiceClient client,
    string accessKeyId,
    string userName,
    string policyArn,
    string roleName)
{
    var detachPolicyResponse = await client.DetachRolePolicyAsync(new
DetachRolePolicyRequest
{
    PolicyArn = policyArn,
    RoleName = roleName,
});
```

```
        var delPolicyResponse = await client.DeletePolicyAsync(new
DeletePolicyRequest
{
    PolicyArn = policyArn,
});

var delRoleResponse = await client.DeleteRoleAsync(new
DeleteRoleRequest
{
    RoleName = roleName,
});

var delAccessKey = await client.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
{
    AccessKeyId = accessKeyId,
    UserName = userName,
});

var delUserResponse = await client.DeleteUserAsync(new
DeleteUserRequest
{
    UserName = userName,
});

}
```

- For API details, see [DetachRolePolicy](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::DetachRolePolicyRequest detachRequest;
detachRequest.SetRoleName(roleName);
detachRequest.SetPolicyArn(policyArn);

auto detachOutcome = iam.DetachRolePolicy(detachRequest);
if (!detachOutcome.IsSuccess()) {
    std::cerr << "Failed to detach policy " << policyArn << " from role "
        << roleName << ":" << detachOutcome.GetError().GetMessage() <<
        std::endl;
}
else {
    std::cout << "Successfully detached policy " << policyArn << " from role "
        << roleName << std::endl;
}

return detachOutcome.IsSuccess();
```

- For API details, see [DetachRolePolicy](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detachPolicy(IamClient iam, String roleName, String policyArn ) {

    try {
        DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.detachRolePolicy(request);
        System.out.println("Successfully detached policy " + policyArn +
            " from role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DetachRolePolicy](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Detach the policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import {
    ListAttachedRolePoliciesCommand,
    DetachRolePolicyCommand,
} from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { RoleName: "ROLE_NAME" }; //ROLE_NAME
```

```
export const run = async () => {
  try {
    const data = await iamClient.send(
      new ListAttachedRolePoliciesCommand(params)
    );
    const myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (_val, index) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        try {
          await iamClient.send(
            new DetachRolePolicyCommand(paramsRoleList)
          );
          console.log("Policy detached from role successfully");
          process.exit();
        } catch (err) {
          console.log("Unable to detach policy from role", err);
        }
      } else {
      });
    });
    return data;
  } catch (err) {
    console.log("User " + "USER_NAME" + " does not exist.");
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DetachRolePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var paramsRoleList = {
  RoleName: process.argv[2]
};

iam.listAttachedRolePolicies(paramsRoleList, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === 'AmazonDynamoDBFullAccess') {
        var params = {
          PolicyArn: 'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess',
          RoleName: process.argv[2]
        };
        iam.detachRolePolicy(params, function(err, data) {
          if (err) {
            console.log("Unable to detach policy from role", err);
          }
        });
      }
    });
  }
});
```

```
        } else {
            console.log("Policy detached from role successfully");
            process.exit();
        }
    });
});
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DetachRolePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun detachPolicy(roleNameVal: String, policyArnVal: String) {

    val request = DetachRolePolicyRequest {
        roleName = roleNameVal
        policyArn = policyArnVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role
$roleNameVal")
    }
}
```

- For API details, see [DetachRolePolicy](#) in [AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Detach a policy from a role using the Boto3 Policy object.

```
def detach_from_role(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """

    session = Session()
    iam = session.client('iam')
    iam.detach_role_policy(RoleName=role_name, PolicyArn=policy_arn)
```

```
"""
try:
    iam.Policy(policy_arn).detach_role(RoleName=role_name)
    logger.info("Detached policy %s from role %s.", policy_arn, role_name)
except ClientError:
    logger.exception(
        "Couldn't detach policy %s from role %s.", policy_arn, role_name)
    raise
```

Detach a policy from a role using the Boto3 Role object.

```
def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name)
        raise
```

- For API details, see [DetachRolePolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a role. If the role has policies attached, they are detached and
# deleted before the role is deleted.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_role(role)
    role.attached_policies.each do |policy|
        name = policy.policy_name
        policy.detach_role(role_name: role.name)
        policy.delete
        puts("Deleted policy #{name}.")
    end
    name = role.name
    role.delete
    puts("Deleted role #{name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't detach policies and delete role #{role.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
end
```

- For API details, see [DetachRolePolicy](#) in *AWS SDK for Ruby API Reference*.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn detach_role_policy(
    client: &iamClient,
    role_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_role_policy()
        .role_name(role_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- For API details, see [DetachRolePolicy](#) in *AWS SDK for Rust API reference*.

## Detach an IAM policy from a user using an AWS SDK

The following code examples show how to detach an IAM policy from a user.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMDetachRolePolicyAPI defines the interface for the DetachRolePolicy function.
// We use this interface to test the function using a mocked service.
type IAMDetachRolePolicyAPI interface {
    DetachRolePolicy(ctx context.Context,
        params *iam.DetachRolePolicyInput,
        optFns ...func(*iam.Options)) (*iam.DetachRolePolicyOutput, error)
}
```

```
// DetachDynamoFullPolicy detaches an Amazon DynamoDB full-access policy from an
// AWS Identity and Access Management (IAM) role.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If successful, a DetachRolePolicyOutput object containing the result of the
//   service call and nil.
//   Otherwise, nil and an error from the call to DetachRolePolicy.
func DetachDynamoFullPolicy(c context.Context, api IAMDetachRolePolicyAPI, input
    *iam.DetachRolePolicyInput) (*iam.DetachRolePolicyOutput, error) {
    return api.DetachRolePolicy(c, input)
}

func main() {
    roleName := flag.String("r", "", "The name of the IAM role")
    flag.Parse()

    if *roleName == "" {
        fmt.Println("You must supply a role name (-r ROLE)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    policyArn := "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess"
    input := &iam.DetachRolePolicyInput{
        PolicyArn: &policyArn,
        RoleName:  roleName,
    }

    _, err = DetachDynamoFullPolicy(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Unable to detach DynamoDB full-access role policy from role")
        return
    }
    fmt.Println("Role detached successfully")
}
```

- For API details, see [DetachUserPolicy](#) in [AWS SDK for Go API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
```

```
"""
try:
    iam.User(user_name).detach_policy(PolicyArn=policy_arn)
    logger.info("Detached policy %s from user %s.", policy_arn, user_name)
except ClientError:
    logger.exception(
        "Couldn't detach policy %s from user %s.", policy_arn, user_name)
    raise
```

- For API details, see [DetachUserPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn detach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- For API details, see [DetachUserPolicy](#) in *AWS SDK for Rust API reference*.

## Generate a credential report from IAM using an AWS SDK

The following code example shows how to generate a credential report from IAM for the current account. After the report is generated, get it by using the `GetCredentialReport` action.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
```

```
    to get the latest report. A new report can be generated a minimum of four hours
    after the last one was generated.
    """
try:
    response = iam.meta.client.generate_credential_report()
    logger.info("Generating credentials report for your account. "
                "Current state is %s.", response['State'])
except ClientError:
    logger.exception("Couldn't generate a credentials report for your
account.")
    raise
else:
    return response
```

- For API details, see [GenerateCredentialReport](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a credential report from IAM using an AWS SDK

The following code example shows how to get the most recently generated credential report from IAM.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_credential_report():
    """
    Gets the most recently generated credentials report about the current account.

    :return: The credentials report.
    """
try:
    response = iam.meta.client.get_credential_report()
    logger.debug(response['Content'])
except ClientError:
    logger.exception("Couldn't get credentials report.")
    raise
else:
    return response['Content']
```

- For API details, see [GetCredentialReport](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a detailed IAM authorization report for your account using an AWS SDK

The following code example shows how to get a detailed IAM authorization report for your account.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report, such
                           as users or roles. When not specified, all resources
                           are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

- For API details, see [GetAccountAuthorizationDetails](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get an IAM policy using an AWS SDK

The following code examples show how to get an IAM policy.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();
var request = new GetPolicyRequest
{
    PolicyArn = "POLICY_ARN",
};

var response = await client.GetPolicyAsync(request);

Console.WriteLine($"{response.Policy.PolicyName} was created on ");
Console.WriteLine($"{response.Policy.CreateDate}");
```

- For API details, see [GetPolicy](#) in *AWS SDK for .NET API Reference*.

C++

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::getPolicy(const Aws::String &policyArn,
                             const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetPolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.GetPolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error getting policy " << policyArn << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &policy = outcome.GetResult().GetPolicy();
        std::cout << "Name: " << policy.GetPolicyName() << std::endl <<
            "ID: " << policy.GetPolicyId() << std::endl << "Arn: " <<
            policy.GetArn() << std::endl << "Description: " <<
            policy.GetDescription() << std::endl << "CreateDate: " <<

        policy.GetCreateDate().ToGmtString(Aws::Utils::DateFormat::ISO_8601)
            << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [GetPolicy](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// GetPolicy

getPolicyResponse, err := service.GetPolicy(context.Background(),
&iam.GetPolicyInput{
    PolicyArn: policyArn,
})

if err != nil {
    panic("Couldn't get policy from ARN: " + err.Error())
}

fmt.Printf("policy: %s, name %s\n",
    *getPolicyResponse.Policy.Arn,
    *getPolicyResponse.Policy.PolicyName)
```

- For API details, see [GetPolicy](#) in *AWS SDK for Go API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Get the policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { GetPolicyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const params = {
  PolicyArn: "POLICY_ARN" /* required */,
};

const run = async () => {
  try {
    const data = await iamClient.send(new GetPolicyCommand(params));
    console.log("Success", data.Policy);
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetPolicy](#) in *AWS SDK for JavaScript API Reference*.

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  PolicyArn: 'arn:aws:iam::aws:policy/AWSLambdaExecute'
```

```
};

iam.getPolicy(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Policy.Description);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetPolicy](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getIAMPolicy(policyArnVal: String?) {

    val request = GetPolicyRequest {
        policyArn = policyArnVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- For API details, see [GetPolicy](#) in [AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

public function getPolicy($policyArn)
{
    return $this->customWaiter(function () use ($policyArn) {
        return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
    });
}
```

- For API details, see [GetPolicy](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_default_policy_statement(policy_arn):  
    """  
    Gets the statement of the default version of the specified policy.  
  
    :param policy_arn: The ARN of the policy to look up.  
    :return: The statement of the default policy version.  
    """  
    try:  
        policy = iam.Policy(policy_arn)  
        # To get an attribute of a policy, the SDK first calls get_policy.  
        policy_doc = policy.default_version.document  
        policy_statement = policy_doc.get('Statement', None)  
        logger.info("Got default policy doc for %s.", policy.policy_name)  
        logger.info(policy_doc)  
    except ClientError:  
        logger.exception("Couldn't get default policy statement for %s.",  
policy_arn)  
        raise  
    else:  
        return policy_statement
```

- For API details, see [GetPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Gets data about a policy.  
#  
# @param policy_arn [String] The ARN of the policy to look up.  
# @return [Aws::IAM::Policy] The retrieved policy.  
def get_policy(policy_arn)  
    policy = @iam_resource.policy(policy_arn)  
    puts("Got policy '#{policy.policy_name}'. Its ID is: #{policy.policy_id}.")  
rescue Aws::Errors::ServiceError => e  
    puts("Couldn't get policy '#{policy_arn}'. Here's why:")  
    puts("\t#{e.code}: #{e.message}")  
    raise  
else  
    policy  
end
```

- For API details, see [GetPolicy](#) in *AWS SDK for Ruby API Reference*.

Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func getPolicy(arn: String) async throws -> IamClientTypes.Policy {  
    let input = GetPolicyInput(  
        policyArn: arn  
    )  
    do {  
        let output = try await client.getPolicy(input: input)  
        guard let policy = output.policy else {  
            throw ServiceHandlerError.noSuchPolicy  
        }  
        return policy  
    } catch {  
        throw error  
    }  
}
```

- For API details, see [GetPolicy](#) in *AWS SDK for Swift API reference*.

## Get an IAM policy version using an AWS SDK

The following code example shows how to get an IAM policy version.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_default_policy_statement(policy_arn):  
    """  
    Gets the statement of the default version of the specified policy.  
  
    :param policy_arn: The ARN of the policy to look up.  
    :return: The statement of the default policy version.  
    """  
    try:  
        policy = iam.Policy(policy_arn)  
        # To get an attribute of a policy, the SDK first calls get_policy.  
        policy_doc = policy.default_version.document  
        policy_statement = policy_doc.get('Statement', None)  
        logger.info("Got default policy doc for %s.", policy.policy_name)  
        logger.info(policy_doc)  
    except ClientError:  
        logger.exception("Couldn't get default policy statement for %s.",  
                         policy_arn)  
        raise  
    else:
```

```
    return policy_statement
```

- For API details, see [GetPolicyVersion](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get an IAM role using an AWS SDK

The following code examples show how to get an IAM role.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

var response = await client.GetRoleAsync(new GetRoleRequest
{
    RoleName = "LambdaS3Role",
});

if (response.Role != null)
{
    Console.WriteLine($"{response.Role.RoleName} with ARN: {response.Role.Arn}");
    Console.WriteLine($"{response.Role.Description}");
    Console.WriteLine($"Created: {response.Role.CreateDate} Last used on:
{ response.Role.RoleLastUsed}");
}
```

- For API details, see [GetRole](#) in *AWS SDK for .NET API Reference*.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// GetRole

getRoleResult, err := service.GetRole(context.Background(), &iam.GetRoleInput{
    RoleName: aws.String(ExampleRoleName),
})

if err != nil {
    panic("Couldn't get role! " + err.Error())
}
```

```
fmt.Println("## GetRole results: ")
fmt.Println("ARN: ", *getRoleResult.Role.Arn)
fmt.Println("Name: ", *getRoleResult.Role.RoleName)
fmt.Println("Created On: ", *getRoleResult.Role.CreateDate)
```

- For API details, see [GetRole](#) in *AWS SDK for Go API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Get the role.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { GetRoleCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const params = {
  RoleName: "ROLE_NAME" /* required */
};

const run = async () => {
  try {
    const data = await iamClient.send(new GetRoleCommand(params));
    console.log("Success", data.Role);
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [GetRole](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

    public function getRole($roleName)
    {
        return $this->customWaiter(function () use ($roleName) {
            return $this->iamClient->getRole(['RoleName' => $roleName]);
        });
    }
```

- For API details, see [GetRole](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_role(role_name):
    """
    Gets a role by name.

    :param role_name: The name of the role to retrieve.
    :return: The specified role.
    """
    try:
        role = iam.Role(role_name)
        role.load() # calls GetRole to load attributes
        logger.info("Got role with arn %s.", role.arn)
    except ClientError:
        logger.exception("Couldn't get role named %s.", role_name)
        raise
    else:
        return role
```

- For API details, see [GetRole](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Gets data about a role.
#
# @param name [String] The name of the role to look up.
# @return [Aws::IAM::Role] The retrieved role.
def get_role(name)
    role = @iam_resource.role(name)
    puts("Got data for role '#{role.name}'. Its ARN is '#{role.arn}'")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't get data for role '#{name}' Here's why:")
    puts("\t#{e.code}: #{e.message}")
```

```
raise
else
  role
end
```

- For API details, see [GetRole](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn get_role(
    client: &iamClient,
    role_name: String,
) -> Result<GetRoleOutput, SdkError<GetRoleError>> {
    let response = client.get_role().role_name(role_name).send().await?;
    Ok(response)
}
```

- For API details, see [GetRole](#) in *AWS SDK for Rust API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func getRole(name: String) async throws -> IamClientTypes.Role {
    let input = GetRoleInput(
        roleName: name
    )
    do {
        let output = try await client.getRole(input: input)
        guard let role = output.role else {
            throw ServiceHandlerError.noSuchRole
        }
        return role
    } catch {
        throw error
    }
}
```

- For API details, see [GetRole](#) in *AWS SDK for Swift API reference*.

## Get an IAM server certificate using an AWS SDK

The following code examples show how to get an IAM server certificate.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::getServerCertificate(const Aws::String &certificateName,
                                         const Aws::Client::ClientConfiguration
                                         &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    auto outcome = iam.GetServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
            Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error getting server certificate " << certificateName <<
                ":" << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                  << "' not found." << std::endl;
        }
    }
    else {
        const auto &certificate = outcome.GetResult().GetServerCertificate();
        std::cout << "Name: " <<

        certificate.GetServerCertificateMetadata().GetServerCertificateName()
            << std::endl << "Body: " << certificate.GetCertificateBody() <<
            std::endl << "Chain: " << certificate.GetCertificateChain() <<
            std::endl;
    }
}

return result;
}
```

- For API details, see [GetServerCertificate](#) in *AWS SDK for C++ API Reference*.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
```

```
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Get a server certificate.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { GetServerCertificateCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { ServerCertificateName: "CERTIFICATE_NAME" }; // CERTIFICATE_NAME

export const run = async () => {
  try {
    const data = await iamClient.send(new GetServerCertificateCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetServerCertificate](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.getServerCertificate({ServerCertificateName: 'CERTIFICATE_NAME'}, function(err,
  data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetServerCertificate](#) in [AWS SDK for JavaScript API Reference](#).

## Get a summary of account usage from IAM using an AWS SDK

The following code example shows how to get a summary of account usage from IAM.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map
```

- For API details, see [GetAccountSummary in AWS SDK for Python \(Boto3\) API Reference](#).

## Get data about the last use of an IAM access key using an AWS SDK

The following code examples show how to get data about the last use of an IAM access key.

### C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::accessKeyLastUsed(const Aws::String &secretKeyID,
                                      const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetAccessKeyLastUsedRequest request;

    request.SetAccessKeyId(secretKeyID);

    Aws::IAM::Model::GetAccessKeyLastUsedOutcome outcome =
iam.GetAccessKeyLastUsed(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error querying last used time for access key " <<
        secretKeyID << ":" << outcome.GetError().GetMessage() <<
std::endl;
    }
    else {
        Aws::String lastUsedTimeString =
        outcome.GetResult()
            .GetAccessKeyLastUsed()
```

```
        .GetLastUsedDate()
        .ToGmtString(Aws::Utils::DateFormat::ISO_8601);
    std::cout << "Access key " << secretKeyID << " last used at time " <<
        lastUsedTimeString << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [GetAccessKeyLastUsed](#) in *AWS SDK for C++ API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMGetAccessKeyLastUsedAPI defines the interface for the GetAccessKeyLastUsed
// function.
// We use this interface to test the function using a mocked service.
type IAMGetAccessKeyLastUsedAPI interface {
    GetAccessKeyLastUsed(ctx context.Context,
        params *iam.GetAccessKeyLastUsedInput,
        optFns ...func(*iam.Options)) (*iam.GetAccessKeyLastUsedOutput, error)
}

// WhenWasKeyUsed retrieves when an AWS Identity and Access Management (IAM) access
// key was last used, including the AWS Region and with which service.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, a GetAccessKeyLastUsedOutput object containing the result of
//     the service call and nil.
//     Otherwise, nil and an error from the call to GetAccessKeyLastUsed.
func WhenWasKeyUsed(c context.Context, api IAMGetAccessKeyLastUsedAPI, input
    *iam.GetAccessKeyLastUsedInput) (*iam.GetAccessKeyLastUsedOutput, error) {
    return api.GetAccessKeyLastUsed(c, input)
}

func main() {
    keyID := flag.String("k", "", "The ID of the access key")
    flag.Parse()

    if *keyID == "" {
        fmt.Println("You must supply the ID of an access key (-k KEY-ID)")
        return
    }
}
```

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic("configuration error, " + err.Error())
}

client := iam.NewFromConfig(cfg)

input := &iam.GetAccessKeyLastUsedInput{
    AccessKeyId: keyID,
}

result, err := WhenWasKeyUsed(context.TODO(), client, input)
if err != nil {
    fmt.Println("Got an error retrieving when access key was last used:")
    fmt.Println(err)
    return
}

fmt.Println("The key was last used:", *result.AccessKeyLastUsed)
```

- For API details, see [GetAccessKeyLastUsed](#) in *AWS SDK for Go API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Get the access key.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { GetAccessKeyLastUsedCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { AccessKeyId: "ACCESS_KEY_ID" }; //ACCESS_KEY_ID

export const run = async () => {
    try {
        const data = await iamClient.send(new GetAccessKeyLastUsedCommand(params));
        console.log("Success", data);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetAccessKeyLastUsed](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.getAccessKeyLastUsed({AccessKeyId: 'ACCESS_KEY_ID'}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.AccessKeyLastUsed);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetAccessKeyLastUsed](#) in [AWS SDK for JavaScript API Reference](#).

## Python

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response['AccessKeyLastUsed'].get('LastUsedDate', None)
        last_service = response['AccessKeyLastUsed'].get('ServiceName', None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.", key_id,
            response['UserName'], last_used_date, last_service)
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response
```

- For API details, see [GetAccessKeyLastUsed](#) in [AWS SDK for Python \(Boto3\) API Reference](#).

## Get the IAM account password policy using an AWS SDK

The following code examples show how to get the IAM account password policy.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

try
{
    var request = new GetAccountPasswordPolicyRequest();
    var response = await client.GetAccountPasswordPolicyAsync(request);

    Console.WriteLine($"{response.PasswordPolicy}");
}
catch (NoSuchEntityException ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}
```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for .NET API Reference*.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// GetAccountPasswordPolicy

fmt.Println("# GetAccountPasswordPolicy")

accountPasswordPolicy, err :=
service.GetAccountPasswordPolicy(context.Background(),
&iam.GetAccountPasswordPolicy{})

if err != nil {
    var notexists *types.NoSuchEntityException
    if errors.As(err, &notexists) {
        fmt.Println("No password policy")
    } else {
        panic("Couldn't get account password policy! " + err.Error())
    }
} else {
    fmt.Println("Users can change password: ",
accountPasswordPolicy.PasswordPolicy.AllowUsersToChangePassword)
```

```
    fmt.Println("Passwords expire: ",
accountPasswordPolicy.PasswordPolicy.ExpirePasswords)
    fmt.Println("Minimum password length: ",
accountPasswordPolicy.PasswordPolicy.MinimumPasswordLength)
}
```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for Go API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Get the account password policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { GetAccountPasswordPolicyCommand } from "@aws-sdk/client-iam";

const run = async () => {
  try {
    const data = await iamClient.send(new GetAccountPasswordPolicyCommand({}));
    console.log("Success", data.PasswordPolicy);
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

public function getAccountPasswordPolicy()
{
```

```
        return $this->iamClient->getAccountPasswordPolicy();  
    }  
}
```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def print_password_policy():  
    """  
    Prints the password policy for the account.  
    """  
    try:  
        pw_policy = iam.AccountPasswordPolicy()  
        print("Current account password policy:")  
        print(f"\tallow_users_to_change_password:  
{pw_policy.allow_users_to_change_password}")  
        print(f"\texpire_passwords: {pw_policy.expire_passwords}")  
        print(f"\thard_expiry: {pw_policy.hard_expiry}")  
        print(f"\tmax_password_age: {pw_policy.max_password_age}")  
        print(f"\tminimum_password_length: {pw_policy.minimum_password_length}")  
        print(f"\tpassword_reuse_prevention:  
{pw_policy.password_reuse_prevention}")  
        print(f"\trequire_lowercase_characters:  
{pw_policy.require_lowercase_characters}")  
        print(f"\trequire_numbers: {pw_policy.require_numbers}")  
        print(f"\trequire_symbols: {pw_policy.require_symbols}")  
        print(f"\trequire_uppercase_characters:  
{pw_policy.require_uppercase_characters}")  
        printed = True  
    except ClientError as error:  
        if error.response['Error']['Code'] == 'NoSuchEntity':  
            print("The account does not have a password policy set.")  
        else:  
            logger.exception("Couldn't get account password policy.")  
            raise  
    else:  
        return printed
```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Prints the password policy for the account.  
def print_account_password_policy  
  policy = @iam_resource.account_password_policy  
  policy.load
```

```
    puts("The account password policy is:")
    puts(policy.data.to_h)
rescue Aws::Errors::ServiceError => e
  if e.code == "NoSuchEntity"
    puts("The account does not have a password policy.")
  else
    puts("Couldn't print the account password policy. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn get_account_password_policy(
    client: &iامClient,
) -> Result<GetAccountPasswordPolicyOutput,
    SdkError<GetAccountPasswordPolicyError>> {
    let response = client.get_account_password_policy().send().await?;

    Ok(response)
}
```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for Rust API reference*.

## List SAML providers for IAM using an AWS SDK

The following code examples show how to list SAML providers for IAM.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

var response = await client.ListSAMLProvidersAsync(new ListSAMLProvidersRequest());
```

```
response.SAMLProviderList.ForEach(samlProvider =>
{
    Console.WriteLine($"{samlProvider.Arn} created on: {samlProvider.CreateDate}");
});
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for .NET API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// ListSAMLProviders

samlProviderList, err := service.ListSAMLProviders(context.Background(),
&iam.ListSAMLProvidersInput{})

if err != nil {
    panic("Couldn't list saml providers: " + err.Error())
}

for _, provider := range samlProviderList.SAMLProviderList {
    fmt.Printf("%s %s -> %s", *provider.Arn, *provider.CreateDate,
*provider.ValidUntil)
}
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for Go API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the SAML providers.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { ListSAMLProvidersCommand } from "@aws-sdk/client-iam";

export const run = async () => {
```

```
try {
    const results = await iamClient.send(new ListSAMLProvidersCommand({}));
    console.log("Success", results);
    return results;
} catch (err) {
    console.log("Error", err);
}
run();
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

public function listSAMLProviders()
{
    return $this->iamClient->listSAMLProviders();
}
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_saml_providers(count):
    """
    Lists the SAML providers for the account.

    :param count: The maximum number of providers to list.
    """
    try:
        found = 0
        for provider in iam.saml_providers.limit(count):
            logger.info('Got SAML provider %s.', provider.arn)
            found += 1
        if found == 0:
            logger.info("Your account has no SAML providers.")
    except ClientError:
        logger.exception("Couldn't list SAML providers.")
        raise
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Lists up to a specified number of SAML providers for the account.  
#  
# @param count [Integer] The maximum number of providers to list.  
def list_saml_providers(count)  
    @iam_resource.saml_providers.limit(count).each do |provider|  
        puts("\t#{provider.arn}")  
    end  
rescue Aws::Errors::ServiceError => e  
    puts("Couldn't list SAML providers. Here's why:")  
    puts("\t#{e.code}: #{e.message}")  
    raise  
end
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_saml_providers(  
    client: &Client,  
) -> Result<ListSamlProvidersOutput, SdkError<ListSAMLProvidersError>> {  
    let response = client.list_saml_providers().send().await?;  
  
    Ok(response)  
}
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for Rust API reference*.

## List a user's IAM access keys using an AWS SDK

The following code examples show how to list a user's IAM access keys.

### C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::listAccessKeys(const Aws::String &userName,
                                  const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccessKeysRequest request;
    request.SetUserName(userName);

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccessKeys(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list access keys for user " << userName
                  << ":" << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "UserName" <<
                std::setw(30) << "KeyID" << std::setw(20) << "Status" <<
                std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &keys = outcome.GetResult().GetAccessKeyMetadata();
        const Aws::String DATE_FORMAT = "%Y-%m-%d";

        for (const auto &key: keys) {
            Aws::String statusString =
                Aws::IAM::Model::StatusTypeMapper::GetNameForStatusType(
                    key.GetStatus());
            std::cout << std::left << std::setw(32) << key.GetUserName() <<
                std::setw(30) << key.GetAccessKeyId() << std::setw(20) <<
                statusString << std::setw(20) <<
                key.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
            std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- For API details, see [ListAccessKeys](#) in [AWS SDK for C++ API Reference](#).

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main
```

```
import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMListAccessKeysAPI defines the interface for the ListAccessKeys function.
// We use this interface to test the function using a mocked service.
type IAMListAccessKeysAPI interface {
    ListAccessKeys(ctx context.Context,
        params *iam.ListAccessKeysInput,
        optFns ...func(*iam.Options)) (*iam.ListAccessKeysOutput, error)
}

// GetAccessKeys retrieves up to the AWS Identity and Access Management (IAM)
// access keys for a user.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, a ListAccessKeysOutput object containing the result of the
//     service call and nil.
//     Otherwise, nil and an error from the call to ListAccessKeys.
func GetAccessKeys(c context.Context, api IAMListAccessKeysAPI, input
    *iam.ListAccessKeysInput) (*iam.ListAccessKeysOutput, error) {
    return api.ListAccessKeys(c, input)
}

func main() {
    maxItems := flag.Int("m", 10, "The maximum number of access keys to show")
    userName := flag.String("u", "", "The name of the user")
    flag.Parse()

    if *userName == "" {
        fmt.Println("You must supply the name of a user (-u USER)")
        return
    }

    if *maxItems < 0 {
        *maxItems = 10
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.ListAccessKeysInput{
        MaxItems: aws.Int32(int32(*maxItems)),
        UserName: userName,
    }

    result, err := GetAccessKeys(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error retrieving user access keys:")
        fmt.Println(err)
        return
    }
}
```

```
        for _, key := range result.AccessKeyMetadata {
            fmt.Println("Status for access key " + *key.AccessKeyId + ": " +
string(key.Status))
        }
    }
```

- For API details, see [ListAccessKeys](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listKeys( IamClient iam, String userName ){

    try {
        boolean done = false;
        String newMarker = null;

        while (!done) {
            ListAccessKeysResponse response;

            if(newMarker == null) {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .build();

                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .marker(newMarker)
                    .build();

                response = iam.listAccessKeys(request);
            }

            for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                System.out.format("Retrieved access key %s",
metadata.accessKeyId());
            }

            if (!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListAccessKeys](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the access keys.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { ListAccessKeysCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  MaxItems: 5,
  UserName: "IAM_USER_NAME", //IAM_USER_NAME
};

export const run = async () => {
  try {
    const data = await iamClient.send(new ListAccessKeysCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListAccessKeys](#) in *AWS SDK for JavaScript API Reference*.

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
```

```
    MaxItems: 5,
    UserName: 'IAM_USER_NAME'
};

iam.listAccessKeys(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListAccessKeys](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listKeys(userNameVal: String?) {

    val request = ListAccessKeysRequest {
        userName = userNameVal
    }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- For API details, see [ListAccessKeys](#) in [AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
```

```
    keys = list(iam.User(user_name).access_keys.all())
    logger.info("Got %s access keys for %s.", len(keys), user_name)
except ClientError:
    logger.exception("Couldn't get access keys for %s.", user_name)
    raise
else:
    return keys
```

- For API details, see [ListAccessKeys](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a user. If the user has inline policies or access keys, they are
# deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
    user.policies.each do |policy|
        name = policy.name
        policy.delete
        puts("Deleted user policy #{name}.")
    end
    user.access_keys.each do |key|
        key.delete
        puts("Deleted access key for user #{user.name}.")
    end
    name = user.name
    user.delete
    puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
end
```

- For API details, see [ListAccessKeys](#) in *AWS SDK for Ruby API Reference*.

## List IAM account aliases using an AWS SDK

The following code examples show how to list IAM account aliases.

### C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool
AwsDoc::IAM::listAccountAliases(const Aws::Client::ClientConfiguration
&clientConfig) {
```

```
Aws::IAM::IAMClient iam(clientConfig);
Aws::IAM::Model::ListAccountAliasesRequest request;

bool done = false;
bool header = false;
while (!done) {
    auto outcome = iam.ListAccountAliases(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to list account aliases: " <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    const auto &aliases = outcome.GetResult().GetAccountAliases();
    if (!header) {
        if (aliases.size() == 0) {
            std::cout << "Account has no aliases" << std::endl;
            break;
        }
        std::cout << std::left << std::setw(32) << "Alias" << std::endl;
        header = true;
    }

    for (const auto &alias: aliases) {
        std::cout << std::left << std::setw(32) << alias << std::endl;
    }

    if (outcome.GetResult().GetIsTruncated()) {
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else {
        done = true;
    }
}

return true;
}
```

- For API details, see [ListAccountAliases](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMListAccountAliasesAPI defines the interface for the ListAccountAliases
function.
```

```
// We use this interface to test the function using a mocked service.
type IAMListAccountAliasesAPI interface {
    ListAccountAliases(ctx context.Context,
        params *iam.ListAccountAliasesInput,
        optFns ...func(*iam.Options)) (*iam.ListAccountAliasesOutput, error)
}

// GetAccountAliases retrieves the aliases for your AWS Identity and Access
// Management (IAM) account.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, a ListAccountAliasesOutput object containing the result of
//     the service call and nil.
//     Otherwise, nil and an error from the call to ListAccountAliases.
func GetAccountAliases(c context.Context, api IAMListAccountAliasesAPI, input
    *iam.ListAccountAliasesInput) (*iam.ListAccountAliasesOutput, error) {
    return api.ListAccountAliases(c, input)
}

func main() {
    maxItems := flag.Int("m", 10, "Maximum number of aliases to list")
    flag.Parse()

    if *maxItems < 0 {
        *maxItems = 10
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.ListAccountAliasesInput{
        MaxItems: aws.Int32(int32(*maxItems)),
    }

    result, err := GetAccountAliases(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error retrieving account aliases")
        fmt.Println(err)
        return
    }

    for i, alias := range result.AccountAliases {
        fmt.Printf("Alias %d: %s\n", i, alias)
    }
}
```

- For API details, see [ListAccountAliases](#) in *AWS SDK for Go API Reference*.

Java

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAliases(IamClient iam) {  
  
    try {  
        ListAccountAliasesResponse response = iam.listAccountAliases();  
        for (String alias : response.accountAliases()) {  
            System.out.printf("Retrieved account alias %s", alias);  
        }  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListAccountAliases](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";  
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

List the account aliases.

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "./libs/iamClient.js";  
import { ListAccountAliasesCommand } from "@aws-sdk/client-iam";  
  
// Set the parameters.  
export const params = { MaxItems: 5 };  
  
export const run = async () => {  
    try {  
        const data = await iamClient.send(new ListAccountAliasesCommand(params));  
        console.log("Success", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListAccountAliases](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.listAccountAliases({MaxItems: 10}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListAccountAliases](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAliases() {

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- For API details, see [ListAccountAliases](#) in [AWS SDK for Kotlin API reference](#).

## Python

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
    alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response['AccountAliases']
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ','.join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response['AccountAliases']
```

- For API details, see [ListAccountAliases](#) in *AWS SDK for Python (Boto3) API Reference*.

## List IAM groups using an AWS SDK

The following code examples show how to list IAM groups.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

var request = new ListGroupsRequest
{
    MaxItems = 10,
};

var response = await client.ListGroupsAsync(request);

do
{
    response.Groups.ForEach(group =>
    {
        Console.WriteLine($"{group.GroupName} created on: {group.CreateDate}");
    });

    if (response.IsTruncated)
    {
        request.Marker = response.Marker;
        response = await client.ListGroupsAsync(request);
    }
} while (response.IsTruncated);
```

- For API details, see [ListGroups](#) in *AWS SDK for .NET API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// ListGroups

listGroupsResult, err := service.ListGroups(context.Background(),
&iam.ListGroupsInput{})

if err != nil {
    panic("Couldn't list groups! " + err.Error())
}

for _, group := range listGroupsResult.Groups {
    fmt.Printf("group %s - %s\n", *group.GroupId, *group.Arn)
}
```

- For API details, see [ListGroups](#) in *AWS SDK for Go API Reference*.

JavaScript

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the groups.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import {ListGroupsCommand} from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
    RoleName: 'ROLE_NAME', /* This is a number value. Required */
    Marker: 'MARKER', /* This is a string value. Optional */
    MaxItems: 'MAX_ITEMS' /* This is a number value. Optional */
```

```
};

export const run = async () => {
  try {
    const data = await iamClient.send(new ListGroupsCommand({}));
    console.log("Success", data.Groups);
  } catch (err) {
    console.log("Error", err);
  }
}
run();
```

- For API details, see [ListGroups](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)
{
    $listGroupsArguments = [];
    if ($pathPrefix) {
        $listGroupsArguments["PathPrefix"] = $pathPrefix;
    }
    if ($marker) {
        $listGroupsArguments["Marker"] = $marker;
    }
    if ($maxItems) {
        $listGroupsArguments["MaxItems"] = $maxItems;
    }

    return $this->iامClient->listGroups($listGroupsArguments);
}
```

- For API details, see [ListGroups](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_groups(count):
    """
    Lists the specified number of groups for the account.

    :param count: The number of groups to list.
    """
```

```
try:
    for group in iam.groups.limit(count):
        logger.info("Group: %s", group.name)
except ClientError:
    logger.exception("Couldn't list groups for the account.")
    raise
```

- For API details, see [ListGroups](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Lists up to a specified number of groups for the account.
#
# @param count [Integer] The maximum number of groups to list.
def list_groups(count)
    @iam_resource.groups.limit(count).each do |group|
        puts("\t#{group.name}")
    end
rescue Aws::Errors::ServiceError => e
    puts("Couldn't list groups for the account. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
end
```

- For API details, see [ListGroups](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_groups(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListGroupsOutput, SdkError<ListGroupsError>> {
    let response = client
        .list_groups()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
```

```
        Ok(response)
    }
```

- For API details, see [ListGroups](#) in *AWS SDK for Rust API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func listGroups() async throws -> [String] {
    var groupList: [String] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListGroupsInput(marker: marker)
        let output = try await client.listGroups(input: input)

        guard let groups = output.groups else {
            return groupList
        }

        for group in groups {
            if let name = group.groupName {
                groupList.append(name)
            }
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return groupList
}
```

- For API details, see [ListGroups](#) in *AWS SDK for Swift API reference*.

## List inline policies for an IAM role using an AWS SDK

The following code examples show how to list inline policies for an IAM role.

### .NET

#### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;
using System;

var client = new AmazonIdentityManagementServiceClient();
var request = new ListRolePoliciesRequest
{
    RoleName = "LambdaS3Role",
};

var response = new ListRolePoliciesResponse();

do
{
    response = await client.ListRolePoliciesAsync(request);

    if (response.PolicyNames.Count > 0)
    {
        response.PolicyNames.ForEach(policyName =>
        {
            Console.WriteLine($"{policyName}");
        });
    }

    // As long as response.IsTruncated is true, set request.Marker equal
    // to response.Marker and call ListRolesAsync again.
    if (response.IsTruncated)
    {
        request.Marker = response.Marker;
    }
} while (response.IsTruncated);
```

- For API details, see [ListRolePolicies](#) in *AWS SDK for .NET API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// ListRolePolicies

rolePoliciesList, err := service.ListRolePolicies(context.Background(),
&iamp;.ListRolePoliciesInput{
    RoleName: aws.String(ExampleRoleName),
})

if err != nil {
    panic("Couldn't list policies for role: " + err.Error())
}

for _, rolePolicy := range rolePoliciesList.PolicyNames {
    fmt.Printf("Policy ARN: %v", rolePolicy)
}
```

- For API details, see [ListRolePolicies](#) in *AWS SDK for Go API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the policies.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import {ListRolePoliciesCommand} from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
    RoleName: 'ROLE_NAME', /* This is a number value. Required */
    Marker: 'MARKER', /* This is a string value. Optional */
    MaxItems: 'MAX_ITEMS' /* This is a number value. Optional */
};

export const run = async () => {
    try {
        const results = await iamClient.send(new ListRolePoliciesCommand(params));
        console.log("Success", results);
        return results;
    } catch (err) {
        console.log("Error", err);
    }
}
run();
```

- For API details, see [ListRolePolicies](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
 getService();

 public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
{
    $listRolePoliciesArguments = ['RoleName' => $roleName];
    if ($marker) {
        $listRolePoliciesArguments['Marker'] = $marker;
    }
}
```

```
        if ($maxItems) {
            $listRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->customWaiter(function () use ($listRolePoliciesArguments) {
            return $this->iamClient->listRolePolicies($listRolePoliciesArguments);
        });
    }
```

- For API details, see [ListRolePolicies](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_policies(role_name):
    """
    Lists inline policies for a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.policies.all():
            logger.info("Got inline policy %s.", policy.name)
    except ClientError:
        logger.exception("Couldn't list inline policies for %s.", role_name)
        raise
```

- For API details, see [ListRolePolicies](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_role_policies(
    client: &iamClient,
    role_name: &str,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolePoliciesOutput, SdkError<ListRolePoliciesError>> {
    let response = client
        .list_role_policies()
        .role_name(role_name)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
```

```
        .await?;

    Ok(response)
}
```

- For API details, see [ListRolePolicies](#) in *AWS SDK for Rust API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func listRolePolicies(role: String) async throws -> [String] {
    var policyList: [String] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListRolePoliciesInput(
            marker: marker,
            roleName: role
        )
        let output = try await client.listRolePolicies(input: input)

        guard let policies = output.policyNames else {
            return policyList
        }

        for policy in policies {
            policyList.append(policy)
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return policyList
}
```

- For API details, see [ListRolePolicies](#) in *AWS SDK for Swift API reference*.

## List IAM policies using an AWS SDK

The following code examples show how to list IAM policies.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;
using System;

var client = new AmazonIdentityManagementServiceClient();

var request = new ListPoliciesRequest
{
    MaxItems = 10,
};

var response = new ListPoliciesResponse();

do
{
    response = await client.ListPoliciesAsync(request);
    response.Policies.ForEach(policy =>
    {
        Console.WriteLine($"{policy.PolicyName} ");
        Console.WriteLine($"with ID: {policy.PolicyId} ");
        Console.WriteLine($"and ARN: {policy.Arn}. ");
        Console.WriteLine($"It was created on {policy.CreateDate}.");
    });

    if (response.IsTruncated)
    {
        request.Marker = response.Marker;
    }
} while (response.IsTruncated);
```

- For API details, see [ListPolicies](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::listPolicies(const Aws::Client::ClientConfiguration
&clientConfig) {
    const Aws::String DATE_FORMAT("%Y-%m-%d");
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListPoliciesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListPolicies(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam policies: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                std::setw(64) << "Description" << std::setw(12) <<
                "CreateDate" << std::endl;
```

```
        header = true;
    }

    const auto & policies = outcome.GetResult().GetPolicies();
    for (const auto & policy: policies) {
        std::cout << std::left << std::setw(55) <<
            policy.GetPolicyName() << std::setw(30) <<
            policy.GetPolicyId() << std::setw(80) << policy.GetArn() <<
            std::setw(64) << policy.GetDescription() << std::setw(12) <<
            policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
            std::endl;
    }

    if (outcome.GetResult().GetIsTruncated()) {
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else {
        done = true;
    }
}

return true;
}
```

- For API details, see [ListPolicies](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// ListPolicies

policyListResponse, err := service.ListPolicies(context.Background(),
&iam.ListPoliciesInput{})

if err != nil {
    panic("Couldn't get list of policies! " + err.Error())
}

fmt.Println("PolicyName\tARN")
for _, policy := range policyListResponse.Policies {
    fmt.Printf("%s\t%s\n", *policy.PolicyName, *policy.Arn)
}
```

- For API details, see [ListPolicies](#) in *AWS SDK for Go API Reference*.

JavaScript

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the policies.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import {ListPoliciesCommand} from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
    Marker: 'MARKER',
    MaxItems: 'MAX_ITEMS',
    OnlyAttached: "ONLY_ATTACHED", /* Options are "true" or "false"*/
    PathPrefix: 'PATH_PREFIX',
    PolicyUsageFilter: "POLICY_USAGE_FILTER", /* Options are "PermissionsPolicy" or
"PermissionsBoundary"*/
    Scope: "SCOPE" /* Options are "All", "AWS", "Local"*/
};

export const run = async () => {
    try {
        const results = await iamClient.send(new ListPoliciesCommand(params));
        console.log("Success", results);
        return results;
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For API details, see [ListPolicies in AWS SDK for JavaScript API Reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
{
    $listPoliciesArguments = [];
    if ($pathPrefix) {
        $listPoliciesArguments["PathPrefix"] = $pathPrefix;
    }
    if ($marker) {
        $listPoliciesArguments["Marker"] = $marker;
    }
}
```

```
        if ($maxItems) {
            $listPoliciesArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listPolicies($listPoliciesArguments);
    }
}
```

- For API details, see [ListPolicies in AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are
                  returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies
```

- For API details, see [ListPolicies in AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Lists up to a specified number of policies in the account.
#
# @param count [Integer] The maximum number of policies to list.
# @return [Array] The Amazon Resource Names (ARNs) of the policies listed.
def list_policies(count)
    policy_arns = []
    @iam_resource.policies.limit(count).each_with_index do |policy, index|
        puts("\t#{index + 1}: #{policy.policy_name}: #{policy.arn}")
        policy_arns.append(policy.arn)
    end
    rescue Aws::Errors::ServiceError => e
        puts("Couldn't list policies for the account. Here's why:")
        puts("\t#{e.code}: #{e.message}")
```

```
    raise
  else
    policy_arns
  end
```

- For API details, see [ListPolicies](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_policies(
    client: &iامClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListPoliciesOutput, SdkError<ListPoliciesError>> {
    let response = client
        .list_policies()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- For API details, see [ListPolicies](#) in *AWS SDK for Rust API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func listPolicies() async throws -> [MyPolicyRecord] {
    var policyList: [MyPolicyRecord] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListPoliciesInput(marker: marker)
        let output = try await client.listPolicies(input: input)
```

```
        guard let policies = output.policies else {
            return policyList
        }

        for policy in policies {
            guard let name = policy.policyName,
                  let id = policy.policyId,
                  let arn = policy.arn else {
                throw ServiceHandlerError.noSuchPolicy
            }
            policyList.append(MyPolicyRecord(name: name, id: id, arn: arn))
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return policyList
}
```

- For API details, see [ListPolicies](#) in *AWS SDK for Swift API reference*.

## List policies attached to an IAM role using an AWS SDK

The following code examples show how to list policies attached to an IAM role.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();
var request = new ListAttachedRolePoliciesRequest
{
    MaxItems = 10,
    RoleName = "testAssumeRole",
};

var response = await client.ListAttachedRolePoliciesAsync(request);

do
{
    response.AttachedPolicies.ForEach(policy =>
    {
        Console.WriteLine($"{policy.PolicyName} with ARN: {policy.PolicyArn}");
    });

    if (response.IsTruncated)
    {
        request.Marker = response.Marker;
        response = await client.ListAttachedRolePoliciesAsync(request);
    }
} while (response.IsTruncated);
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for .NET API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// ListAttachedRolePolicies

attachedPoliciesList, err :=
    service.ListAttachedRolePolicies(context.Background(),
    &iam.ListAttachedRolePoliciesInput{
        RoleName: aws.String(ExampleRoleName),
    })

if err != nil {
    panic("Couldn't call ListAttachedRolePolicies: " + err.Error())
}

fmt.Println("## List attached role policies for " + ExampleRoleName)

for _, attachedPolicy := range attachedPoliciesList.AttachedPolicies {
    fmt.Printf("attached policy: %v\n (%v) \n", attachedPolicy.PolicyArn,
    attachedPolicy.PolicyName)
}
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for Go API Reference*.

JavaScript

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the policies that are attached to a role.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import {ListAttachedRolePoliciesCommand} from "@aws-sdk/client-iam";

// Set the parameters.
```

```
export const params = {
  RoleName: 'ROLE_NAME' /* required */
};

export const run = async () => {
  try {
    const data = await iamClient.send(new
ListAttachedRolePoliciesCommand(params));
    console.log("Success", data.AttachedPolicies);
  } catch (err) {
    console.log("Error", err);
  }
}
run();
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker =
"", $maxItems = 0)
{
    $listAttachRolePoliciesArguments = ['RoleName' => $roleName];
    if ($pathPrefix) {
        $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;
    }
    if ($marker) {
        $listAttachRolePoliciesArguments['Marker'] = $marker;
    }
    if ($maxItems) {
        $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;
    }
    return $this->iامClient-
>listAttachedRolePolicies($listAttachRolePoliciesArguments);
}
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_attached_policies(role_name):
    """
    Lists policies attached to a role.
```

```
:param role_name: The name of the role to query.  
"""  
try:  
    role = iam.Role(role_name)  
    for policy in role.attached_policies.all():  
        logger.info("Got policy %s.", policy.arn)  
except ClientError:  
    logger.exception("Couldn't list attached policies for %s.", role_name)  
    raise
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a role. If the role has policies attached, they are detached and  
# deleted before the role is deleted.  
#  
# @param role [Aws::IAM::Role] The role to delete.  
def delete_role(role)  
    role.attached_policies.each do |policy|  
        name = policy.policy_name  
        policy.detach_role(role_name: role.name)  
        policy.delete  
        puts("Deleted policy #{name}.")  
    end  
    name = role.name  
    role.delete  
    puts("Deleted role #{name}.")  
rescue Aws::Errors::ServiceError => e  
    puts("Couldn't detach policies and delete role #{role.name}. Here's why:")  
    puts("\t#{e.code}: #{e.message}")  
    raise  
end
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_attached_role_policies(  
    client: &iamClient,  
    role_name: String,
```

```
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListAttachedRolePoliciesOutput,
SdkError<ListAttachedRolePoliciesError>> {
    let response = client
        .list_attached_role_policies()
        .role_name(role_name)
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for Rust API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// Returns a list of AWS Identity and Access Management (IAM) policies
/// that are attached to the role.
///
/// - Parameter role: The IAM role to return the policy list for.
///
/// - Returns: An array of `IamClientTypes.AttachedPolicy` objects
///   describing each managed policy that's attached to the role.
public func listAttachedRolePolicies(role: String) async throws ->
[IamClientTypes.AttachedPolicy] {
    var policyList: [IamClientTypes.AttachedPolicy] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListAttachedRolePoliciesInput(
            marker: marker,
            roleName: role
        )
        let output = try await client.listAttachedRolePolicies(input: input)

        guard let attachedPolicies = output.attachedPolicies else {
            return policyList
        }

        for attachedPolicy in attachedPolicies {
            policyList.append(attachedPolicy)
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return policyList
}
```

```
}
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for Swift API reference*.

## List IAM roles using an AWS SDK

The following code examples show how to list IAM roles.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

// Without the MaxItems value, the ListRolesAsync method will
// return information for up to 100 roles. If there are more
// than the MaxItems value or more than 100 roles, the response
// value IsTruncated will be true.
var request = new ListRolesRequest
{
    MaxItems = 10,
};

var response = new ListRolesResponse();

do
{
    response = await client.ListRolesAsync(request);
    response.Roles.ForEach(role =>
    {
        Console.WriteLine($"{role.RoleName} - ARN {role.Arn}");
    });

    // As long as response.IsTruncated is true, set request.Marker equal
    // to response.Marker and call ListRolesAsync again.
    if (response.IsTruncated)
    {
        request.Marker = response.Marker;
    }
} while (response.IsTruncated);
```

- For API details, see [ListRoles](#) in *AWS SDK for .NET API Reference*.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// ListRoles

roles, err := service.ListRoles(context.Background(), &iam.ListRolesInput{})

if err != nil {
    panic("Could not list roles: " + err.Error())
}

fmt.Println("## list roles")
for _, idxRole := range roles.Roles {

    fmt.Printf("%s\t%s\t%s\t",
        *idxRole.RoleId,
        *idxRole.RoleName,
        *idxRole.Arn)
    if idxRole.Description != nil {
        fmt.Print(*idxRole.Description)
    }
    fmt.Println("\n")
}
```

- For API details, see [ListRoles](#) in *AWS SDK for Go API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the roles.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { ListRolesCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const params = {
    Marker: 'MARKER', // This is a string value.
    MaxItems: 'MAX_ITEMS' // This is a number value.
};

const run = async () => {
    try {
        const results = await iamClient.send(new ListRolesCommand(params));
        console.log("Success", results);
        return results;
    } catch (err) {
```

```
        console.log("Error", err);
    };
run();
```

- For API details, see [ListRoles](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

/**
 * @param string $pathPrefix
 * @param string $marker
 * @param int $maxItems
 * @return Result
 * $roles = $service->listRoles();
 */
public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
{
    $listRolesArguments = [];
    if ($pathPrefix) {
        $listRolesArguments["PathPrefix"] = $pathPrefix;
    }
    if ($marker) {
        $listRolesArguments["Marker"] = $marker;
    }
    if ($maxItems) {
        $listRolesArguments["MaxItems"] = $maxItems;
    }
    return $this->iامClient->listRoles($listRolesArguments);
}
```

- For API details, see [ListRoles](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_roles(count):
    """
    Lists the specified number of roles for the account.

    :param count: The number of roles to list.
    """
    try:
        roles = list(iam.roles.limit(count=count))
```

```
        for role in roles:
            logger.info("Role: %s", role.name)
    except ClientError:
        logger.exception("Couldn't list roles for the account.")
        raise
    else:
        return roles
```

- For API details, see [ListRoles](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Lists up to a specified number of roles for the account.
#
# @param count [Integer] The maximum number of roles to list.
# @return [Array] The names of the listed roles.
def list_roles(count)
  role_names = []
  @iam_resource.roles.limit(count).each_with_index do |role, index|
    puts("\t#{index + 1}: #{role.name}")
    role_names.append(role.name)
  end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list roles for the account. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    role_names
  end
```

- For API details, see [ListRoles](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_roles(
    client: &iامClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolesOutput, SdkError<ListRolesError>> {
    let response = client
        .list_roles()
```

```
.set_path_prefix(path_prefix)
.set_marker(marker)
.set_max_items(max_items)
.send()
.await?;
Ok(response)
}
```

- For API details, see [ListRoles](#) in *AWS SDK for Rust API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func listRoles() async throws -> [String] {
    var roleList: [String] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListRolesInput(marker: marker)
        let output = try await client.listRoles(input: input)

        guard let roles = output.roles else {
            return roleList
        }

        for role in roles {
            if let name = role.roleName {
                roleList.append(name)
            }
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return roleList
}
```

- For API details, see [ListRoles](#) in *AWS SDK for Swift API reference*.

## List IAM server certificates using an AWS SDK

The following code examples show how to list IAM server certificates.

### C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::listServerCertificates(
    const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";

    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListServerCertificatesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListServerCertificates(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list server certificates: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                std::setw(30) << std::setw(80) << "Arn" <<
                std::setw(14) << "UploadDate" << std::setw(14) <<
                "ExpirationDate" << std::endl;
            header = true;
        }

        const auto &certificates =
            outcome.GetResult().GetServerCertificateMetadataList();

        for (const auto &certificate: certificates) {
            std::cout << std::left << std::setw(55) <<
                certificate.GetServerCertificateName() << std::setw(30) <<
                certificate.GetServerCertificateId() << std::setw(80) <<
                certificate.GetArn() << std::setw(14) <<
                certificate.GetUploadDate().ToGmtString(DATE_FORMAT.c_str())
            <<
                std::setw(14) <<
                certificate.GetExpiration().ToGmtString(DATE_FORMAT.c_str())
            <<
                std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- For API details, see [ListServerCertificates in AWS SDK for C++ API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the certificates.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { ListServerCertificatesCommand } from "@aws-sdk/client-iam";

export const run = async () => {
  try {
    const data = await iamClient.send(new ListServerCertificatesCommand({}));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListServerCertificates in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.listServerCertificates({}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListServerCertificates in AWS SDK for JavaScript API Reference](#).

## List IAM users using an AWS SDK

The following code examples show how to list IAM users.

.NET

**AWS SDK for .NET**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();
var request = new ListUsersRequest
{
    MaxItems = 10,
};
var response = await client.ListUsersAsync(request);

do
{
    response.Users.ForEach(user =>
    {
        Console.WriteLine($"{{user.UserName}} created on {{user.CreateDate}}.");
        Console.WriteLine($"ARN: {{user.Arn}}\n");
    });

    request.Marker = response.Marker;
    response = await client.ListUsersAsync(request);
} while (response.IsTruncated);
```

- For API details, see [ListUsers](#) in *AWS SDK for .NET API Reference*.

C++

**SDK for C++**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::listUsers(const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListUsersRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListUsers(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam users:" <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "Name" <<
```

```
        std::setw(30) << "ID" << std::setw(64) << "Arn" <<
        std::setw(20) << "CreateDate" << std::endl;
    header = true;
}

const auto &users = outcome.GetResult().GetUsers();
for (const auto &user: users) {
    std::cout << std::left << std::setw(32) << user.GetUserName() <<
    std::setw(30) << user.GetUserId() << std::setw(64) <<
    user.GetArn() << std::setw(20) <<
    user.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
    << std::endl;
}

if (outcome.GetResult().GetIsTruncated()) {
    request.SetMarker(outcome.GetResult().GetMarker());
}
else {
    done = true;
}
}

return true;
}
```

- For API details, see [ListUsers](#) in *AWS SDK for C++ API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// ListUsers

fmt.Println("## List users")

userListResult, err := service.ListUsers(context.Background(),
&iam.ListUsersInput{})
if err != nil {
    panic("Couldn't list users: " + err.Error())
}
for _, userResult := range userListResult.Users {
    fmt.Printf("%s\t%s\n", *userResult.UserName, *userResult.Arn)
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllUsers(IamClient iam) {  
  
    try {  
  
        boolean done = false;  
        String newMarker = null;  
  
        while(!done) {  
            ListUsersResponse response;  
            if (newMarker == null) {  
                ListUsersRequest request = ListUsersRequest.builder().build();  
                response = iam.listUsers(request);  
            } else {  
                ListUsersRequest request = ListUsersRequest.builder()  
                    .marker(newMarker)  
                    .build();  
  
                response = iam.listUsers(request);  
            }  
  
            for(User user : response.users()) {  
                System.out.format("\n Retrieved user %s", user.userName());  
                AttachedPermissionsBoundary permissionsBoundary =  
user.permissionsBoundary();  
                if (permissionsBoundary != null)  
                    System.out.format("\n Permissions boundary details %s",  
permissionsBoundary.permissionsBoundaryAsString());  
            }  
  
            if(!response.isTruncated()) {  
                done = true;  
            } else {  
                newMarker = response.marker();  
            }  
        }  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";  
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

List the users.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { ListUsersCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { MaxItems: 10 };

export const run = async () => {
  try {
    const data = await iamClient.send(new ListUsersCommand(params));
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListUsers](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  MaxItems: 10
};

iam.listUsers(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var users = data.Users || [];
    users.forEach(function(user) {
      console.log("User " + user.UserName + " created", user.CreateDate);
    });
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListUsers](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAllUsers() {  
  
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        val response = iamClient.listUsers(ListUsersRequest { })  
        response.users?.forEach { user ->  
            println("Retrieved user ${user.userName}")  
            val permissionsBoundary = user.permissionsBoundary  
            if (permissionsBoundary != null)  
                println("Permissions boundary details  
${permissionsBoundary.permissionsBoundaryType}")  
        }  
    }  
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();  
$service = new IamService();  
  
public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)  
{  
    $listUsersArguments = [];  
    if ($pathPrefix) {  
        $listUsersArguments["PathPrefix"] = $pathPrefix;  
    }  
    if ($marker) {  
        $listUsersArguments["Marker"] = $marker;  
    }  
    if ($maxItems) {  
        $listUsersArguments["MaxItems"] = $maxItems;  
    }  
  
    return $this->iamClient->listUsers($listUsersArguments);  
}
```

- For API details, see [ListUsers](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users
```

- For API details, see [ListUsers](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Lists up to a specified number of users in the account.
#
# @param count [Integer] The maximum number of users to list.
def list_users(count)
  @iam_resource.users.limit(count).each do |user|
    puts("\t#{user.name}")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't list users for the account. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- For API details, see [ListUsers](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_users(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListUsersOutput, SdkError<ListUsersError>> {
    let response = client
        .list_users()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Rust API reference*.

Swift

**SDK for Swift**

**Note**

This is prerelease documentation for an SDK in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func listUsers() async throws -> [MyUserRecord] {
    var userList: [MyUserRecord] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListUsersInput(marker: marker)
        let output = try await client.listUsers(input: input)

        guard let users = output.users else {
            return userList
        }

        for user in users {
            if let id = user.userId, let name = user.userName {
                userList.append(MyUserRecord(id: id, name: name))
            }
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return userList
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Swift API reference*.

## Update an IAM server certificate using an AWS SDK

The following code examples show how to update an IAM server certificate.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::updateServerCertificate(const Aws::String
    &currentCertificateName,
                                              const Aws::String &newCertificateName,
                                              const Aws::Client::ClientConfiguration
    &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateServerCertificateRequest request;
    request.SetServerCertificateName(currentCertificateName);
    request.SetNewServerCertificateName(newCertificateName);

    auto outcome = iam.UpdateServerCertificate(request);
    bool result = true;
    if (outcome.IsSuccess()) {
        std::cout << "Server certificate " << currentCertificateName
            << " successfully renamed as " << newCertificateName
            << std::endl;
    }
    else {
        if (outcome.GetError().GetErrorCode() !=
            Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error changing name of server certificate " <<
                currentCertificateName << " to " << newCertificateName << ":" <<
                outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << currentCertificateName
                << "' not found." << std::endl;
        }
    }
    return result;
}
```

- For API details, see [UpdateServerCertificate](#) in *AWS SDK for C++ API Reference*.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
```

```
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

Update a server certificate.

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "./libs/iamClient.js";  
import { UpdateServerCertificateCommand } from "@aws-sdk/client-iam";  
  
// Set the parameters.  
export const params = {  
    ServerCertificateName: "CERTIFICATE_NAME", //CERTIFICATE_NAME  
    NewServerCertificateName: "NEW_CERTIFICATE_NAME", //NEW_CERTIFICATE_NAME  
};  
  
export const run = async () => {  
    try {  
        const data = await iamClient.send(  
            new UpdateServerCertificateCommand(params)  
        );  
        console.log("Success", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateServerCertificate](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create the IAM service object  
var iam = new AWS.IAM({apiVersion: '2010-05-08'});  
  
var params = {  
    ServerCertificateName: 'CERTIFICATE_NAME',  
    NewServerCertificateName: 'NEW_CERTIFICATE_NAME'  
};  
  
iam.updateServerCertificate(params, function(err, data) {  
    if (err) {  
        console.log("Error", err);  
    } else {  
        console.log("Success", data);  
    }  
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateServerCertificate](#) in *AWS SDK for JavaScript API Reference*.

## Update an IAM user using an AWS SDK

The following code examples show how to update an IAM user.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::updateUser(const Aws::String &currentUserName,
                               const Aws::String &newUserName,
                               const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::UpdateUserRequest request;
    request.SetUserName(currentUserName);
    request.SetNewUserName(newUserName);

    auto outcome = iam.UpdateUser(request);
    if (outcome.IsSuccess()) {
        std::cout << "IAM user " << currentUserName <<
                    " successfully updated with new user name " << newUserName <<
                    std::endl;
    }
    else {
        std::cerr << "Error updating user name for IAM user " << currentUserName <<
                    ":" << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [UpdateUser](#) in *AWS SDK for C++ API Reference*.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
```

```

"fmt"

"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMUpdateUserAPI defines the interface for the UpdateUser function.
// We use this interface to test the function using a mocked service.
type IAMUpdateUserAPI interface {
    UpdateUser(ctx context.Context,
        params *iam.UpdateUserInput,
        optFns ...func(*iam.Options)) (*iam.UpdateUserOutput, error)
}

// RenameUser changes the name for an AWS Identity and Access Management (IAM)
// user.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, a UpdateUserOutput object containing the result of the
//     service call and nil.
//     Otherwise, nil and an error from the call to UpdateUser.
func RenameUser(c context.Context, api IAMUpdateUserAPI, input
    *iam.UpdateUserInput) (*iam.UpdateUserOutput, error) {
    return api.UpdateUser(c, input)
}

func main() {
    userName := flag.String("u", "", "The name of the user")
    newName := flag.String("n", "", "The new name of the user")
    flag.Parse()

    if *userName == "" || *newName == "" {
        fmt.Println("You must supply a user name and new name (-u USERNAME -n NEW-NAME)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.UpdateUserInput{
        UserName:    userName,
        NewUserName: newName,
    }

    _, err = RenameUser(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error updating user " + *userName)
    }

    fmt.Println("Updated user name from: " + *userName + " to: " + *newName)
}

```

- For API details, see [UpdateUser in AWS SDK for Go API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void updateIAMUser(IamClient iam, String curName, String newName) {
    try {
        UpdateUserRequest request = UpdateUserRequest.builder()
            .userName(curName)
            .newUserName(newName)
            .build();

        iam.updateUser(request);
        System.out.printf("Successfully updated user to username %s", newName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [UpdateUser in AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Update the user.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { UpdateUserCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
    UserName: "ORIGINAL_USER_NAME", //ORIGINAL_USER_NAME
    NewUserName: "NEW_USER_NAME", //NEW_USER_NAME
};

export const run = async () => {
```

```
try {
    const data = await iamClient.send(new UpdateUserCommand(params));
    console.log("Success, username updated");
    return data;
} catch (err) {
    console.log("Error", err);
}
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateUser in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
    UserName: process.argv[2],
    NewUserName: process.argv[3]
};

iam.updateUser(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateUser in AWS SDK for JavaScript API Reference](#).

## Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun updateIAMUser(curName: String?, newName: String?) {  
    val request = UpdateUserRequest {
```

```
        userName = curName
        newUserName = newName
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

- For API details, see [UpdateUser in AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user
```

- For API details, see [UpdateUser in AWS SDK for Python \(Boto3\) API Reference](#).

## Update an IAM access key using an AWS SDK

The following code examples show how to update an IAM access key.

### C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::updateAccessKey(const Aws::String &userName,
                                    const Aws::String &accessKeyId,
                                    Aws::IAM::Model::StatusType status,
                                    const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
```

```
Aws::IAM::Model::UpdateAccessKeyRequest request;
request.SetUserName(userName);
request.SetAccessKeyId(accessKeyID);
request.setStatus(status);

auto outcome = iam.UpdateAccessKey(request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully updated status of access key "
        << accessKeyID << " for user " << userName << std::endl;
}
else {
    std::cerr << "Error updated status of access key " << accessKeyID <<
        " for user " << userName << ":" <<
        outcome.GetError().GetMessage() << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [UpdateAccessKey](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// IAMUpdateAccessKeyAPI defines the interface for the UpdateAccessKey function.
// We use this interface to test the function using a mocked service.
type IAMUpdateAccessKeyAPI interface {
    UpdateAccessKey(ctx context.Context,
        params *iam.UpdateAccessKeyInput,
        optFns ...func(*iam.Options)) (*iam.UpdateAccessKeyOutput, error)
}

// ActivateKey sets the status of an AWS Identity and Access Management (IAM)
// access key to active.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If successful, a UpdateAccessKeyOutput object containing the result of the
//   service call and nil.
//   Otherwise, nil and an error from the call to UpdateAccessKey.
func ActivateKey(c context.Context, api IAMUpdateAccessKeyAPI, input
    *iam.UpdateAccessKeyInput) (*iam.UpdateAccessKeyOutput, error) {
    return api.UpdateAccessKey(c, input)
```

```
}

func main() {
    keyID := flag.String("k", "", "The ID of the access key")
    userName := flag.String("u", "", "The name of the user")
    flag.Parse()

    if *keyID == "" || *userName == "" {
        fmt.Println("You must supply an access key ID and user name (-k KEY-ID -u USERNAME)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.UpdateAccessKeyInput{
        AccessKeyId: keyID,
        Status:       types.StatusTypeActive,
        UserName:     userName,
    }

    _, err = ActivateKey(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Error", err)
        return
    }

    fmt.Println("Access Key activated")
}
```

- For API details, see [UpdateAccessKey](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void updateKey(IamClient iam, String username, String accessId,
String status ) {

    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
            statusType = StatusType.INACTIVE;
        } else {
            statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
        }

        UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
            .accessKeyId(accessId)
            .userName(username)
            .status(statusType)
```

```
        .build();

        iam.updateAccessKey(request);
        System.out.printf("Successfully updated the status of access key %s to"
+
                "status %s for user %s", accessId, status, username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [UpdateAccessKey](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Update the access key.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { UpdateAccessKeyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
    AccessKeyId: "ACCESS_KEY_ID", //ACCESS_KEY_ID
    Status: "Active",
    UserName: "USER_NAME", //USER_NAME
};

export const run = async () => {
    try {
        const data = await iamClient.send(new UpdateAccessKeyCommand(params));
        console.log("Success", data);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateAccessKey](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  AccessKeyId: 'ACCESS_KEY_ID',
  Status: 'Active',
  UserName: 'USER_NAME'
};

iam.updateAccessKey(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateAccessKey](#) in [AWS SDK for JavaScript API Reference](#).

## Python

### SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
    deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", 'Activated' if activate else 'Deactivated',
key_id)
    except ClientError:
        logger.exception()
```

```
key_id)      "Couldn't %s key %s.", 'Activate' if activate else 'Deactivate',
            raise
```

- For API details, see [UpdateAccessKey](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios for IAM using AWS SDKs

The following code examples show how to use AWS Identity and Access Management with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Create an IAM user and assume a role with AWS STS using an AWS SDK \(p. 1027\)](#)
- [Create read-only and read-write IAM users using an AWS SDK \(p. 1078\)](#)
- [Manage IAM access keys using an AWS SDK \(p. 1084\)](#)
- [Manage IAM policies using an AWS SDK \(p. 1087\)](#)
- [Manage IAM roles using an AWS SDK \(p. 1090\)](#)
- [Manage your IAM account using an AWS SDK \(p. 1093\)](#)
- [Roll back an IAM policy version using an AWS SDK \(p. 1097\)](#)

## Create an IAM user and assume a role with AWS STS using an AWS SDK

The following code examples show how to:

- Create a user who has no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list Amazon S3 buckets using temporary credentials.
- Delete the policy, role, and user.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;
using Amazon.S3;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;

public class IAM_Basics
```

```
{
    // Values needed for user, role, and policies.
    private const string UserName = "example-user";
    private const string S3PolicyName = "s3-list-buckets-policy";
    private const string RoleName = "temporary-role";
    private const string AssumePolicyName = "sts-trust-user";

    private static readonly RegionEndpoint Region = RegionEndpoint.USEast2;

    public static async Task Main()
    {
        DisplayInstructions();

        // Create the IAM client object.
        var client = new AmazonIdentityManagementServiceClient(Region);

        // First create a user. By default, the new user has
        // no permissions.
        Console.WriteLine($"Creating a new user with user name: {UserName}.");
        var user = await CreateUserAsync(client, UserName);
        var userArn = user.Arn;
        Console.WriteLine($"Successfully created user: {UserName} with ARN:
{userArn}.");

        // Create an AccessKey for the user.
        var accessKey = await CreateAccessKeyAsync(client, UserName);

        var accessKeyId = accessKey.AccessKeyId;
        var secretAccessKey = accessKey.SecretAccessKey;

        // Try listing the Amazon Simple Storage Service (Amazon S3)
        // buckets. This should fail at this point because the user doesn't
        // have permissions to perform this task.
        var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
        await ListMyBucketsAsync(s3Client1);

        // Define a role policy document that allows the new user
        // to assume the role.
        // string assumeRolePolicyDocument =
        File.ReadAllText("assumePolicy.json");
        string assumeRolePolicyDocument = "{" +
            "\\"Version\\": \"2012-10-17\", " +
            "\\"Statement\\": [{" +
                "\\"Effect\\": \"Allow\", " +
                "\\"Principal\\": {" +
                    $" \\"AWS\\": \"{userArn}\\\" " +
                "}, " +
                "\\"Action\\": \"sts:AssumeRole\" " +
            "}] " +
        "}";

        // Permissions to list all buckets.
        string policyDocument = "{" +
            "\\"Version\\": \"2012-10-17\", " +
            " \\"Statement\\\" : [{" +
                " \\"Action\\\" : [\"s3>ListAllMyBuckets\"], " +
                " \\"Effect\\\" : \"Allow\", " +
                " \\"Resource\\\" : \"*\" " +
            "}] " +
        "}";

        // Create the role to allow listing the S3 buckets. Role names are
        // not case sensitive and must be unique to the account for which it
        // is created.
        var role = await CreateRoleAsync(client, RoleName,
        assumeRolePolicyDocument);
}
```

```
        var roleArn = role.Arn;

        // Create a policy with permissions to list S3 buckets
        var policy = await CreatePolicyAsync(client, S3PolicyName,
policyDocument);

        // Wait 15 seconds for the policy to be created.
        WaitABit(15, "Waiting for the policy to be available.");

        // Attach the policy to the role you created earlier.
        await AttachRoleAsync(client, policy.Arn, RoleName);

        // Wait 15 seconds for the role to be updated.
        Console.WriteLine();
        WaitABit(15, "Waiting to time for the policy to be attached.");

        // Use the AWS Security Token Service (AWS STS) to have the user
        // assume the role we created.
        var stsClient = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        // Wait for the new credentials to become valid.
        WaitABit(10, "Waiting for the credentials to be valid.");

        var assumedRoleCredentials = await AssumeS3RoleAsync(stsClient,
"temporary-session", roleArn);

        // Try again to list the buckets using the client created with
        // the new user's credentials. This time, it should work.
        var s3Client2 = new AmazonS3Client(assumedRoleCredentials);

        await ListMyBucketsAsync(s3Client2);

        // Now clean up all the resources used in the example.
        await DeleteResourcesAsync(client, accessKeyId, UserName, policy.Arn,
RoleName);

        Console.WriteLine("IAM Demo completed.");
    }

    ///<summary>
    /// Create a new IAM user.
    ///</summary>
    ///<param name="client">The initialized IAM client object.</param>
    ///<param name="userName">A string representing the user name of the
    /// new user.</param>
    ///<returns>The newly created user.</returns>
    public static async Task<User> CreateUserAsync(
        AmazonIdentityManagementServiceClient client,
        string userName)
    {
        var request = new CreateUserRequest
        {
            UserName = userName,
        };

        var response = await client.CreateUserAsync(request);

        return response.User;
    }

    ///<summary>
    /// Create a new AccessKey for the user.

```

```

    ///> </summary>
    ///> <param name="client">The initialized IAM client object.</param>
    ///> <param name="userName">The name of the user for whom to create the
key.</param>
    ///> <returns>A new IAM access key for the user.</returns>
    public static async Task<AccessKey> CreateAccessKeyAsync(
        AmazonIdentityManagementServiceClient client,
        string userName)
    {
        var request = new CreateAccessKeyRequest
        {
            UserName = userName,
        };

        var response = await client.CreateAccessKeyAsync(request);

        if (response.AccessKey is not null)
        {
            Console.WriteLine($"Successfully created Access Key for
{userName}.");
        }

        return response.AccessKey;
    }

    ///> <summary>
    ///> Create a policy to allow a user to list the buckets in an account.
    ///> </summary>
    ///> <param name="client">The initialized IAM client object.</param>
    ///> <param name="policyName">The name of the policy to create.</param>
    ///> <param name="policyDocument">The permissions policy document.</param>
    ///> <returns>The newly created ManagedPolicy object.</returns>
    public static async Task<ManagedPolicy> CreatePolicyAsync(
        AmazonIdentityManagementServiceClient client,
        string policyName,
        string policyDocument)
    {
        var request = new CreatePolicyRequest
        {
            PolicyName = policyName,
            PolicyDocument = policyDocument,
        };

        var response = await client.CreatePolicyAsync(request);

        return response.Policy;
    }

    ///> <summary>
    ///> Attach the policy to the role so that the user can assume it.
    ///> </summary>
    ///> <param name="client">The initialized IAM client object.</param>
    ///> <param name="policyArn">The ARN of the policy to attach.</param>
    ///> <param name="roleName">The name of the role to attach the policy to.</
param>
    public static async Task AttachRoleAsync(
        AmazonIdentityManagementServiceClient client,
        string policyArn,
        string roleName)
    {
        var request = new AttachRolePolicyRequest
        {

```

```
        PolicyArn = policyArn,
        RoleName = roleName,
    };

    var response = await client.AttachRolePolicyAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine("Successfully attached the policy to the role.");
    }
    else
    {
        Console.WriteLine("Could not attach the policy.");
    }
}

/// <summary>
/// Create a new IAM role which we can attach to a user.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="roleName">The name of the IAM role to create.</param>
/// <param name="rolePermissions">The permissions which the role will
have.</param>
/// <returns>A Role object representing the newly created role.</returns>
public static async Task<Role> CreateRoleAsync(
    AmazonIdentityManagementServiceClient client,
    string roleName,
    string rolePermissions)
{
    var request = new CreateRoleRequest
    {
        RoleName = roleName,
        AssumeRolePolicyDocument = rolePermissions,
    };

    var response = await client.CreateRoleAsync(request);

    return response.Role;
}

/// <summary>
/// List the Amazon S3 buckets owned by the user.
/// </summary>
/// <param name="accessKeyId">The access key Id for the user.</param>
/// <param name="secretAccessKey">The Secret access key for the user.</
param>
public static async Task ListMyBucketsAsync(AmazonS3Client client)
{
    Console.WriteLine("\nPress <Enter> to list the S3 buckets using the new
user.\n");
    Console.ReadLine();

    try
    {
        // Get the list of buckets accessible by the new user.
        var response = await client.ListBucketsAsync();

        // Loop through the list and print each bucket's name
        // and creation date.
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Listing S3 buckets:\n");
        response.Buckets
```

```

        .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName}, 
created on: {b.CreationDate}"));
    }
    catch (AmazonS3Exception ex)
    {
        // Something else went wrong. Display the error message.
        Console.WriteLine($"Error: {ex.Message}");
    }

    Console.WriteLine("Press <Enter> to continue.");
    Console.ReadLine();
}

/// <summary>
/// Have the user assume the role that allows the role to be used to
/// list all S3 buckets.
/// </summary>
/// <param name="client">An initialized AWS STS client object.</param>
/// <param name="roleSession">The name of the session where the role
/// assumption will be active.</param>
/// <param name="roleToAssume">The Amazon Resource Name (ARN) of the
/// role to assume.</param>
/// <returns>The AssumedRoleUser object needed to perform the list
/// buckets procedure.</returns>
public static async Task<Credentials> AssumeS3RoleAsync(
    AmazonSecurityTokenServiceClient client,
    string roleSession,
    string roleToAssume)
{
    // Create the request to use with the AssumeRoleAsync call.
    var request = new AssumeRoleRequest()
    {
        RoleSessionName = roleSession,
        RoleArn = roleToAssume,
    };

    var response = await client.AssumeRoleAsync(request);

    return response.Credentials;
}

/// <summary>
/// Delete the user, and other resources created for this example.
/// </summary>
/// <param name="client">The initialized client object.</param>
/// <param name="accessKeyId">The Id of the user's access key.</param>
/// <param name="userName">The user name of the user to delete.</param>
/// <param name="policyName">The name of the policy to delete.</param>
/// <param name="policyArn">The Amazon Resource Name ARN of the Policy to
delete.</param>
/// <param name="roleName">The name of the role that will be deleted.</param>
public static async Task DeleteResourcesAsync(
    AmazonIdentityManagementServiceClient client,
    string accessKeyId,
    string userName,
    string policyArn,
    string roleName)
{
    var detachPolicyResponse = await client.DetachRolePolicyAsync(new
DetachRolePolicyRequest
{
}

```

```
        PolicyArn = policyArn,
        RoleName = roleName,
    });

    var delPolicyResponse = await client.DeletePolicyAsync(new
DeletePolicyRequest
{
    PolicyArn = policyArn,
});

    var delRoleResponse = await client.DeleteRoleAsync(new
DeleteRoleRequest
{
    RoleName = roleName,
});

    var delAccessKey = await client.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
{
    AccessKeyId = accessKeyId,
    UserName = userName,
});

    var delUserResponse = await client.DeleteUserAsync(new
DeleteUserRequest
{
    UserName = userName,
});

}

/// <summary>
/// Display a countdown and wait for a number of seconds.
/// </summary>
/// <param name="numSeconds">The number of seconds to wait.</param>
public static void WaitABit(int numSeconds, string msg)
{
    Console.WriteLine(msg);

    // Wait for the requested number of seconds.
    for (int i = numSeconds; i > 0; i--)
    {
        System.Threading.Thread.Sleep(1000);
        Console.Write($"{i}...");
    }

    Console.WriteLine("\n\nPress <Enter> to continue.");
    Console.ReadLine();
}

/// <summary>
/// Shows the a description of the features of the program.
/// </summary>
public static void DisplayInstructions()
{
    var separator = new string('-', 80);

    Console.WriteLine(separator);
    Console.WriteLine("IAM Basics");
    Console.WriteLine("This application uses the basic features of the AWS Identity and Access");
    Console.WriteLine("Management (IAM) creating, managing, and controlling access to resources for");
    Console.WriteLine("users. The application was created using the AWS SDK for .NET version 3.7 and");
}
```

```
        Console.WriteLine(".NET Core 5. The application performs the following
actions:");
        Console.WriteLine();
        Console.WriteLine("1. Creates a user with no permissions");
        Console.WriteLine("2. Creates a role and policy that grants
s3>ListAllMyBuckets permission");
        Console.WriteLine("3. Grants the user permission to assume the role");
        Console.WriteLine("4. Creates an Amazon Simple Storage Service (Amazon
S3) client and tries");
        Console.WriteLine("    to list buckets. (This should fail.)");
        Console.WriteLine("5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("6. Creates an Amazon S3 client object with the
temporary credentials and");
        Console.WriteLine("    lists the buckets. (This time it should work.)");
        Console.WriteLine("7. Deletes all of the resources created.");
        Console.WriteLine(separator);
        Console.WriteLine("Press <Enter> to continue.");
        Console.ReadLine();
    }
}
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

C++

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace AwsDoc {
    namespace IAM {
        //! Cleanup by deleting created entities.
        /**
         *sa DeleteCreatedEntities
         *\param client: IAM client.
         *\param role: IAM role.
         *\param user: IAM user.
         *\param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
```

```

        const Aws::IAM::Model::Role &role,
        const Aws::IAM::Model::User &user,
        const Aws::IAM::Model::Policy &policy);
    }

///! Scenario to create an IAM user, create an IAM role, and apply the role to the
// user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
// necessary to
//   create a custom policy).
/*!
 \sa iamCreateUserAssumeRoleScenario
 \param clientConfig: Aws client configuration.
 \return bool: Successful completion.
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
    const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
            Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);

        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            std::cout << "Successfully created IAM user " << userName << std::endl;
        }

        user = outcome.GetResult(). GetUser();
    }

    // 2. Create a role.
    {
        // Get the IAM user for the current client in order to access its ARN.
        Aws::String iamUserArn;
        {
            Aws::IAM::Model:: GetUserRequest request;
            Aws::IAM::Model:: GetUserOutcome outcome = client.GetUser(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error getting Iam user. " <<
                    outcome.GetError().GetMessage() << std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            else {
                std::cout << "Successfully retrieved Iam user "
                    << outcome.GetResult(). GetUser().GetUserName()
                    << std::endl;
            }
        }
    }
}

```

```
        iamUserArn = outcome.GetResult().GetUser().GetArn();
    }

    Aws::IAM::Model::CreateRoleRequest request;

    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String roleName = "iam-demo-role-" +
                           Aws::Utils::StringUtils::ToLower(uuid.c_str());
    request.SetRoleName(roleName);

    // Build policy document for role.
    Aws::Utils::Document jsonStatement;
    jsonStatement.WithString("Effect", "Allow");

    Aws::Utils::Document jsonPrincipal;
    jsonPrincipal.WithString("AWS", iamUserArn);
    jsonStatement.WithObject("Principal", jsonPrincipal);
    jsonStatement.WithString("Action", "sts:AssumeRole");
    jsonStatement.WithObject("Condition", Aws::Utils::Document());

    Aws::Utils::Document policyDocument;
    policyDocument.WithString("Version", "2012-10-17");

    Aws::Utils::Array<Aws::Utils::Document> statements(1);
    statements[0] = jsonStatement;
    policyDocument.WithArray("Statement", statements);

    std::cout << "Setting policy for role\n" <<
        policyDocument.View().WriteCompact() << std::endl;

    // Set role policy document as JSON string.
    request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

    Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating role. " <<
            outcome.GetError().GetMessage() << std::endl;

        DeleteCreatedEntities(client, role, user, policy);
        return false;
    }
    else {
        std::cout << "Successfully created a role with name " << roleName
               << std::endl;
    }

    role = outcome.GetResult().GetRole();
}

// 3. Create an IAM policy.
{
    Aws::IAM::Model::CreatePolicyRequest request;
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String policyName = "iam-demo-policy-" +
                           Aws::Utils::StringUtils::ToLower(uuid.c_str());
    request.SetPolicyName(policyName);

    // Build IAM policy document.
    Aws::Utils::Document jsonStatement;
    jsonStatement.WithString("Effect", "Allow");
    jsonStatement.WithString("Action", "s3>ListAllMyBuckets");
    jsonStatement.WithString("Resource", "arn:aws:s3:::*");

    Aws::Utils::Document policyDocument;
    policyDocument.WithString("Version", "2012-10-17");
```

```
Aws::Utils::Array< Aws::Utils::Document> statements(1);
statements[0] = jsonStatement;
policyDocument.WithArray("Statement", statements);

    std::cout << "Creating a policy.\n" <<
policyDocument.View().WriteCompact()
    << std::endl;

// Set IAM policy document as JSON string.
request.SetPolicyDocument(policyDocument.View().WriteCompact());

Aws::IAM::Model::CreatePolicyOutcome outcome =
client.CreatePolicy(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error creating policy. " <<
        outcome.GetError().GetMessage() << std::endl;

    DeleteCreatedEntities(client, role, user, policy);
    return false;
}
else {
    std::cout << "Successfully created a policy with name, " << policyName
<<
    " ." << std::endl;
}

policy = outcome.GetResult().GetPolicy();
}

// 4. Assume the new role using the AWS Security Token Service (STS).
Aws::STS::Model::Credentials credentials;
{
    Aws::STS::STSCient stsClient(clientConfig);

    Aws::STS::Model::AssumeRoleRequest request;
    request.SetRoleArn(role.GetArn());
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String roleSessionName = "iam-demo-role-session-" +
        Aws::Utils::StringUtils::ToLower(uuid.c_str());
    request.SetRoleSessionName(roleSessionName);

    Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

    // Repeatedly call AssumeRole, because there is often a delay
    // before the role is available to be assumed.
    // Repeat at most 20 times when access is denied.
    int count = 0;
    while (true) {
        assumeRoleOutcome = stsClient.AssumeRole(request);
        if (!assumeRoleOutcome.IsSuccess()) {
            if (count > 20 ||
                assumeRoleOutcome.GetError().GetErrorCode() !=
                Aws::STS::STSErrors::ACCESS_DENIED) {
                std::cerr << "Error assuming role after 20 tries. " <<
                    assumeRoleOutcome.GetError().GetMessage() <<
                std::endl;
                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {
            std::cout << "Successfully assumed the role after " << count
            << " seconds." << std::endl;
        }
    }
}
```

```

        break;
    }
    count++;
}

credentials = assumeRoleOutcome.GetResult().GetCredentials();
}

// 5. List objects in the bucket (This should fail).
{
    Aws::S3::S3Client s3Client(
        Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
            credentials.GetSecretAccessKey(),
            credentials.GetSessionToken()),
        clientConfig);
    Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
    if (!listBucketsOutcome.IsSuccess()) {
        if (listBucketsOutcome.GetError().GetErrorType() !=
            Aws::S3::S3Errors::ACCESS_DENIED) {
            std::cerr << "Could not lists buckets. " <<
                listBucketsOutcome.GetError().GetMessage() << std::endl;
        }
        else {
            std::cout
                << "Access to list buckets denied because privileges have
not been applied."
                << std::endl;
        }
    }
    else {
        std::cerr
            << "Successfully retrieved bucket lists when this should not
happen."
            << std::endl;
    }
}

// 6. Attach the policy to the role.
{
    Aws::IAM::Model::AttachRolePolicyRequest request;
    request.SetRoleName(role.GetRoleName());
    request.WithPolicyArn(policy.GetArn());

    Aws::IAM::Model::AttachRolePolicyOutcome outcome = client.AttachRolePolicy(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy. " <<
            outcome.GetError().GetMessage() << std::endl;

        DeleteCreatedEntities(client, role, user, policy);
        return false;
    }
    else {
        std::cout << "Successfully attached the policy with name, "
            << policy.GetPolicyName() <<
            ", to the role, " << role.GetRoleName() << "." << std::endl;
    }
}

int count = 0;
// 7. List objects in the bucket (this should succeed).
// Repeatedly call ListBuckets, because there is often a delay
// before the policy with ListBucket permissions has been applied to the role.
// Repeat at most 20 times when access is denied.

```

```

        while (true) {
            Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                    credentials.GetSecretAccessKey(),
                    credentials.GetSessionToken()),
                clientConfig);
            Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
            if (!listBucketsOutcome.IsSuccess()) {
                if ((count > 20) ||
                    listBucketsOutcome.GetError().GetErrorType() !=
                    Aws::S3::S3Errors::ACCESS_DENIED) {
                    std::cerr << "Could not lists buckets after 20 seconds. " <<
                        listBucketsOutcome.GetError().GetMessage() << std::endl;
                    DeleteCreatedEntities(client, role, user, policy);
                    return false;
                }
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                std::cout << "Successfully retrieved bucket lists after " << count
                    << " seconds." << std::endl;
                break;
            }
            count++;
        }

        // 8. Delete all the created resources.
        return DeleteCreatedEntities(client, role, user, policy);
    }

    bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                            const Aws::IAM::Model::Role &role,
                                            const Aws::IAM::Model::User &user,
                                            const Aws::IAM::Model::Policy &policy) {
        bool result = true;
        if (policy.ArnsHasBeenSet()) {
            // Detach the policy from the role.
            {
                Aws::IAM::Model::DetachRolePolicyRequest request;
                request.SetPolicyArn(policy.GetArn());
                request.SetRoleName(role.GetRoleName());

                Aws::IAM::Model::DetachRolePolicyOutcome outcome =
client.DetachRolePolicy(
                    request);
                if (!outcome.IsSuccess()) {
                    std::cerr << "Error Detaching policy from roles. " <<
                        outcome.GetError().GetMessage() << std::endl;
                    result = false;
                }
                else {
                    std::cout << "Successfully detached the policy with arn "
                        << policy.GetArn()
                        << " from role " << role.GetRoleName() << "." <<
std::endl;
                }
            }

            // Delete the policy.
            {
                Aws::IAM::Model::DeletePolicyRequest request;
                request.WithPolicyArn(policy.GetArn());

```

```

Aws::IAM::Model::DeletePolicyOutcome outcome =
client.DeletePolicy(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error deleting policy. " <<
        outcome.GetError().GetMessage() << std::endl;
    result = false;
}
else {
    std::cout << "Successfully deleted the policy with arn "
        << policy.GetArn() << std::endl;
}
}

if (role.RoleIdHasBeenSet()) {
// Delete the role.
Aws::IAM::Model::DeleteRoleRequest request;
request.SetRoleName(role.GetRoleName());

Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error deleting role. " <<
        outcome.GetError().GetMessage() << std::endl;
    result = false;
}
else {
    std::cout << "Successfully deleted the role with name "
        << role.GetRoleName() << std::endl;
}
}

if (user.ArnHasBeenSet()) {
// Delete the user.
Aws::IAM::Model::DeleteUserRequest request;
request.WithUserName(user.GetUserName());

Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error deleting user. " <<
        outcome.GetError().GetMessage() << std::endl;
    result = false;
}
else {
    std::cout << "Successfully deleted the user with name "
        << user.GetUserName() << std::endl;
}
}

return result;
}

```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)

- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "errors"
    "fmt"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/credentials"
    "github.com/aws/aws-sdk-go-v2/credentials/stscreds"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/sts"
    "github.com/aws/smithy-go"
)

/*
This is a basic scenario for using AWS Identity and Access Management (IAM) and AWS
Security Token Service (STS).

This example will do the following:

    * Create a user that has no permissions.
    * Create a role and policy that grant s3>ListAllMyBuckets permission.
    * Grant the user permission to assume the role.
    * Create an S3 client object as the user and try to list buckets (this should
fail).
    * Get temporary credentials by assuming the role.
    * Create an S3 client object with the temporary credentials and list the buckets
(this should succeed).
    * Delete all the resources.

*/

const (
    scenario_UserName          = "ExampleUser123"
    scenario_BucketListerRoleName = "BucketListerRole"
    scenario_BucketListerPolicyName = "BucketListerListMyBucketsPolicy"
)

func scenario() {
    cfg, err := config.LoadDefaultConfig(context.Background())
    if err != nil {
        panic("Couldn't load a configuration")
    }
}
```

```
}

iamSvc := iam.NewFromConfig(cfg)

fmt.Println("## Create user")

var userInfo types.User

user, err := iamSvc.CreateUser(context.Background(), &iam.CreateUserInput{
    UserName: aws.String(scenario_UserName),
})

if err != nil {

    var existsAlready *types.EntityAlreadyExistsException
    if errors.As(err, &existsAlready) {
        // The user possibly exists.
        // Check if the user actually exists within IAM.
        user, err := iamSvc.GetUser(context.Background(), &iam.GetUserInput{UserName:
aws.String(scenario_UserName)})
        if err != nil {
            panic("Can't get user: " + err.Error())
        } else {
            // Make sure the user info is set for later.
            userInfo = *user.User
            fmt.Println("User already existed...")
        }
    } else {
        fmt.Println("Couldn't create user! " + err.Error())
    }
} else {
    userInfo = *user.User
}

fmt.Printf("User %s has id %s\n", scenario_UserName, *userInfo.Arn)

fmt.Println("## Create access key")

creds, err := iamSvc.CreateAccessKey(context.Background(),
&iam.CreateAccessKeyInput{
    UserName: aws.String(scenario_UserName),
})

if err != nil {
    panic("Couldn't create credentials for a user! " + err.Error())
}

akId := *creds.AccessKey.AccessKeyId
sakId := *creds.AccessKey.SecretAccessKey

fmt.Printf("## CREDs: accessKeyId(%s) Secretkey(%s)\n", akId, sakId)

// Grant the user the ability to assume the role.

fmt.Println("# waiting for a few moments for keys to become available")

time.Sleep(10 * time.Second)

fmt.Println("## Create the bucket lister role")
bucketListerRole, err := iamSvc.CreateRole(context.Background(),
&iam.CreateRoleInput{
    RoleName: aws.String(scenario_BucketListerRoleName),
    AssumeRolePolicyDocument: aws.String(`{
        "Version": "2012-10-17",
        "Statement": [
            {

```

```
        "Effect": "Allow",
        "Principal": {
            "AWS": ` + (*userInfo.Arn) + `"
        },
        "Action": "sts:AssumeRole"
    }
]
`),
Description: aws.String("Role to let users list their buckets."),
})
var bucketListerRoleArn string
if err != nil {
    // Check to see if the role exists.
    var existsException *types.EntityAlreadyExistsException
    if errors.As(err, &existsException) {
        // Check if we can look up the role as it stands already
        tRole, err := iamSvc.GetRole(context.Background(), &iam.GetRoleInput{
            RoleName: aws.String(scenario_BucketListerRoleName),
        })
        if err != nil {
            // Told it already exists, but now it's gone.
            panic("Couldn't find seemingly extant role: " + err.Error())
        } else {
            bucketListerRoleArn = *tRole.Role.Arn
        }
    } else {
        panic("Couldn't create role! " + err.Error())
    }
} else {
    bucketListerRoleArn = *bucketListerRole.Role.Arn
}

fmt.Printf("## The ARN for the bucket lister role is %s", bucketListerRoleArn)

fmt.Println("## Create policy to allow bucket listing")
bucketAllowPolicy := aws.String(`{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1646154730759",
            "Action": [
                "s3>ListAllMyBuckets"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}`)

var bucketListerPolicyArn string
bucketListerPolicy, err := iamSvc.CreatePolicy(context.Background(),
&iam.CreatePolicyInput{
    PolicyName: aws.String(scenario_BucketListerPolicyName),
    PolicyDocument: bucketAllowPolicy,
    Description: aws.String("Allow user to list their own buckets"),
})
if err != nil {
    var existsException *types.EntityAlreadyExistsException
    if errors.As(err, &existsException) {

        stsClient := sts.NewFromConfig(cfg)
        mCallerId, _ := stsClient.GetCallerIdentity(context.Background(),
&sts.GetCallerIdentityInput{})

        mpolicyArn := fmt.Sprintf("arn:aws:iam::%s:policy/%s", *mCallerId.Account,
scenario_BucketListerPolicyName)

        _, err := iamSvc.GetPolicy(context.Background(), &iam.GetPolicyInput{

```

```
    PolicyArn: &mpolicyArn,
})
if err != nil {
    panic("Failed to find policy by arn(" + mpolicyArn + ") -> " + err.Error())
}
bucketListerPolicyArn = mpolicyArn

} else {
    panic("Couldn't create policy! " + err.Error())
}
} else {
    bucketListerPolicyArn = *bucketListerPolicy.Policy.Arn
}

fmt.Println("## Attach role policy")

-, err = iamSvc.AttachRolePolicy(context.Background(), &iam.AttachRolePolicyInput{
    PolicyArn: &bucketListerPolicyArn,
    RoleName: aws.String(scenario_BucketListerRoleName),
})

if err != nil {
    fmt.Println("Couldn't attach policy to role! " + err.Error())
}

fmt.Printf("# attached role policy ")

fmt.Println("# waiting for a few moments for keys to become available")

time.Sleep(10 * time.Second)

// Create an S3 client that acts as the user.
userConfig, err := config.LoadDefaultConfig(context.TODO(),
    // Use credentials created earlier.
    config.WithCredentialsProvider(credentials.StaticCredentialsProvider{
        Value: aws.Credentials{
            AccessKeyID: akId, SecretAccessKey: sakId,
            Source: "Example user creds",
        },
    }))
if err != nil {
    panic("Couldn't create config for new credentials! " + err.Error())
} else {
    creds, err := userConfig.Credentials.Retrieve(context.Background())
    if err != nil {
        panic("Couldn't get credentials for our new config, something is wrong: " +
            err.Error())
    }
    fmt.Printf("-> config creds are %s %s\n", creds.AccessKeyID,
        creds.SecretAccessKey)
}

s3Client := s3.NewFromConfig(userConfig)
// Attempt to list buckets.
-, err = s3Client.ListBuckets(context.Background(), &s3.ListBucketsInput{})

if err == nil {
    fmt.Println("Call to s3>ListBuckets was not denied (unexpected!)")
} else {
    var oe smithy.APIError
    if errors.As(err, &oe) && (oe.ErrorCode() == "AccessDenied") {
        fmt.Println("Couldn't list buckets (expected!)")
    } else {
        panic("unexpected error: " + err.Error())
    }
}
```

```
stsClient := sts.NewFromConfig(userConfig)
roleCreds := stscreds.NewAssumeRoleProvider(stsClient, bucketListerRoleArn)

tmpCreds, err := roleCreds.Retrieve(context.Background())
if err != nil {
    fmt.Println("couldn't get role creds: " + err.Error())
} else {
    fmt.Printf("role creds: %s %v\n\n", tmpCreds.AccessKeyID, tmpCreds.Expires)
}

roleConfig, err := config.LoadDefaultConfig(context.Background(),
    config.WithCredentialsProvider(roleCreds),
)
if err != nil {
    panic("Couldn't create config with assumed role! " + err.Error())
}

roleS3client := s3.NewFromConfig(roleConfig)

mybuckets, err := roleS3client.ListBuckets(context.Background(),
    &s3.ListBucketsInput{})
if err != nil {
    panic("Couldn't list buckets while assuming role that allows this: " +
        err.Error())
} else {
    fmt.Println("Buckets owned by the user....")
    for _, bucket := range mybuckets.Buckets {
        fmt.Printf("%s -> %s\n", *bucket.Name, bucket.CreationDate)
    }
}
// ---- Clean up ----

// Delete the user's access keys.

fmt.Println("cleanup: Delete created access key")
_, _ = iamSvc.DeleteAccessKey(context.Background(), &iam.DeleteAccessKeyInput{
    AccessKeyId: &akId,
    UserName:    aws.String(scenario_UserName),
})

// Delete the user.
fmt.Println("cleanup: delete the user we created")
_, err = iamSvc.DeleteUser(context.Background(), &iam.DeleteUserInput{UserName:
aws.String(scenario_UserName)})
if err != nil {
    fmt.Println("Couldn't delete user! " + err.Error())
}

// Detach the role policy.

fmt.Println("cleanup: Detach the policy from the role")
_, err = iamSvc.DetachRolePolicy(context.Background(), &iam.DetachRolePolicyInput{
    RoleName: aws.String(scenario_BucketListerRoleName),
    PolicyArn: &bucketListerPolicyArn,
})

if err != nil {
    fmt.Println("Couldn't detach role policy from role " + err.Error())
}

// Delete the role.
fmt.Println("cleanup: Remove the role")
_, err = iamSvc.DeleteRole(context.Background(), &iam.DeleteRoleInput{
```

```
    RoleName: aws.String(scenario_BucketListerRoleName),
})
if err != nil {
    fmt.Println("Couldn't delete role! " + err.Error())
}

// Delete the policy.
fmt.Println("cleanup: delete the policy")
_, err = iamSvc.DeletePolicy(context.Background(), &iam.DeletePolicyInput{
    PolicyArn: &bucketListerPolicyArn,
})
if err != nil {
    fmt.Println("couldn't delete policy!")
}

fmt.Println("done!")
}
```

- For API details, see the following topics in *AWS SDK for Go API Reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap IAM user actions.

```
/*
To run this Java V2 code example, set up your development environment, including
your credentials.

For information, see this documentation topic:

https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

This example performs these operations:

1. Creates a user that has no permissions.
2. Creates a role and policy that grants Amazon S3 permissions.
3. Creates a role.
4. Grants the user permissions.
```

```
    5. Gets temporary credentials by assuming the role. Creates an Amazon S3 Service
client object with the temporary credentials.
    6. Deletes the resources.
*/



public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    public static final String PolicyDocument =
    "{" +
        "  \"Version\": \"2012-10-17\", " +
        "  \"Statement\": [ " +
        "    { " +
        "      \"Effect\": \"Allow\", " +
        "      \"Action\": [ " +
        "        \"s3:*\" " +
        "      ], " +
        "      \"Resource\": \"*\" " +
        "    } " +
        "  ] " +
    "}";
}

public static void main(String[] args) throws Exception {

    final String usage = "\n" +
        "Usage:\n" +
        "  <username> <policyName> <roleName> <roleSessionName>
<fileLocation> <bucketName> \n\n" +
        "Where:\n" +
        "  <username> - The name of the IAM user to create. \n\n" +
        "  <policyName> - The name of the policy to create. \n\n" +
        "  <roleName> - The name of the role to create. \n\n" +
        "  <roleSessionName> - The name of the session required for the
assumeRole operation. \n\n" +
        "  <fileLocation> - The file location to the JSON required to create
the role (see Readme). \n\n" +
        "  <bucketName> - The name of the Amazon S3 bucket from which objects
are read. \n\n" ;

    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }

    String userName = args[0];
    String policyName = args[1];
    String roleName = args[2];
    String roleSessionName = args[3];
    String fileLocation = args[4];
    String bucketName = args[5];

    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS IAM example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println(" 1. Create the IAM user.");
    Boolean createUser = createIAMUser(iam, userName);
    System.out.println(DASHES);
```

```

        if (createUser) {
            System.out.println(userName + " was successfully created.");

            System.out.println(DASHES);
            System.out.println("2. Creates a policy.");
            String polArn = createIAMPolicy(iam, policyName);
            System.out.println("The policy " + polArn + " was successfully
created.");
            System.out.println(DASHES);

            System.out.println(DASHES);
            System.out.println("3. Creates a role.");
            String roleArn = createIAMRole(iam, roleName, fileLocation);
            System.out.println(roleArn + " was successfully created.");
            System.out.println(DASHES);

            System.out.println(DASHES);
            System.out.println("4. Grants the user permissions.");
            attachIAMRolePolicy(iam, roleName, polArn);
            System.out.println(DASHES);

            System.out.println(DASHES);
            System.out.println("**** Wait for 1 MIN so the resource is available");
            TimeUnit.MINUTES.sleep(1);
            System.out.println("5. Gets temporary credentials by assuming the
role.");
            System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
            assumeGivenRole(roleArn, roleSessionName, bucketName);
            System.out.println(DASHES);

            System.out.println(DASHES);
            System.out.println("6 Getting ready to delete the AWS resources");
            deleteRole(iam, roleName, polArn);
            deleteIAMUser(iam, userName);
            System.out.println(DASHES);

            System.out.println(DASHES);
            System.out.println("This IAM Scenario has successfully completed");
            System.out.println(DASHES);
        } else {
            System.out.println(userName +" was not successfully created.");
        }
    }

    public static Boolean createIAMUser(IamClient iam, String username ) {

        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                .userName(username)
                .build();

            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                .userName(response.user().userName())
                .build();

            WaiterResponse< GetUserResponse> waitUntilUserExists =
            iamWaiter.waitUntilUserExists(userRequest);

            waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return true;
        }
    }
}

```

```
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return false;
    }

    public static String createIAMRole(IamClient iam, String rolename, String
fileLocation ) throws Exception {

    try {
        JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(jsonObject.toJSONString())
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is "+response.role().arn());
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createIAMPolicy(IamClient iam, String policyName ) {

    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();
        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);

        // Wait until the policy is created.
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

        waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
            .roleName(roleName)
            .build();
    
```

```
        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        String polArn;
        for (AttachedPolicy policy: attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn)==0) {
                System.out.println(roleName + " policy is already attached to
this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
    .roleName(roleName)
    .policyArn(policyArn)
    .build();

        iam.attachRolePolicy(attachRequest);
        System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Invoke an Amazon S3 operation using the Assumed Role.
public static void assumeGivenRole(String roleArn, String roleSessionName,
String bucketName) {

    StsClient stsClient = StsClient.builder()
    .region(Region.US_EAST_1)
    .build();

    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
    .roleArn(roleArn)
    .roleSessionName(roleSessionName)
    .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials myCreds = roleResponse.credentials();
        String key = myCreds.accessKeyId();
        String secKey = myCreds.secretAccessKey();
        String secToken = myCreds.sessionToken();

        // List all objects in an Amazon S3 bucket using the temp creds.
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()

    .credentialsProvider(StaticCredentialsProvider.create(AwsSessionCredentials.create(key,
secKey, secToken)))
    .region(region)
    .build();

        System.out.println("Created a S3Client using temp credentials.");
        System.out.println("Listing objects in "+bucketName);
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
    .bucket(bucketName)
    .build();

    
```

```
ListObjectsResponse res = s3.listObjects(listObjects);
List<S3Object> objects = res.contents();
for (S3Object myValue : objects) {
    System.out.println("The name of the key is " + myValue.key());
    System.out.println("The owner is " + myValue.owner());
}
}

} catch (StsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void deleteRole(IamClient iam, String roleName, String polArn) {

    try {
        // First the policy needs to be detached.
        DetachRolePolicyRequest rolePolicyRequest =
        DetachRolePolicyRequest.builder()
            .policyArn(polArn)
            .roleName(roleName)
            .build();

        iam.detachRolePolicy(rolePolicyRequest);

        // Delete the policy.
        DeletePolicyRequest request = DeletePolicyRequest.builder()
            .policyArn(polArn)
            .build();

        iam.deletePolicy(request);
        System.out.println("*** Successfully deleted "+polArn);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " +roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteIAMUser(IamClient iam, String userName) {

    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("*** Successfully deleted " + userName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static Object readJsonSimpleDemo(String filename) throws Exception {
    FileReader reader = new FileReader(filename);
    JSONParser jsonParser = new JSONParser();
```

```
        return jsonParser.parse(reader);
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
export const REGION = "REGION"; // For example, "us-east-1".
// Create an Amazon S3 service client object.
export const iamClient = new IAMClient({ region: REGION });
```

Create an IAM user and a role that grants permission to list Amazon S3 buckets. The user has rights only to assume the role. After assuming the role, use temporary credentials to list buckets for the account.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient, REGION } from "../libs/iamClient.js"; // Helper function that
  creates an IAM service client module.
import {
  CreateUserCommand,
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  AttachUserPolicyCommand,
  DeleteAccessKeyCommand,
  DeleteUserCommand,
  DeleteRoleCommand,
  DeletePolicyCommand,
  DetachUserPolicyCommand,
  DetachRolePolicyCommand,
```

```
    } from "@aws-sdk/client-iam";
    import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
    import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";

    if (process.argv.length < 6) {
        console.log(
            "Usage: node iam_basics.js <user name> <s3 policy name> <role name> <assume
            policy name>\n" +
            "Example: node iam_basics.js test-user my-s3-policy my-iam-role my-assume-
            role"
        );
    }
    // Set the parameters.
    const region = REGION;
    const userName = process.argv[2];
    const s3_policy_name = process.argv[3];
    const role_name = process.argv[4];
    const assume_policy_name = process.argv[5];

    // Helper function to delay running the code while the AWS service calls wait for
    // responses.
    function wait(ms) {
        var start = Date.now();
        var end = start
        while (end < start + ms){
            end = Date.now()
        }
    }

    export const run = async (
        userName,
        s3_policy_name,
        role_name,
        assume_policy_name
    ) => {
        try {
            // Create a new user.
            const user_params = { UserName: userName };
            console.log("\nCreating a user name " + user_params.UserName + "...\\n");
            const data = await iamClient.send(
                new CreateUserCommand({ UserName: userName })
            );
            const user_arn = data.User.Arn;
            const user_name = data.User.UserName;
            console.log(
                "User with name" + user_name + " and ARN " + user_arn + " created."
            );
            try {
                // Create access keys for the new user.
                console.log(
                    "\nCreating access keys for " + user_params.UserName + "...\\n"
                );
                const access_key_params = { UserName: user_name };
                const data = await iamClient.send(
                    new CreateAccessKeyCommand(access_key_params)
                );
                console.log("Success. Access key created: ", data.AccessKey.AccessKeyId);
                var myAccessKey = data.AccessKey.AccessKeyId;
                var mySecretAccessKey = data.AccessKey.SecretAccessKey;

                try {
                    // Attempt to list S3 buckets.
                    console.log(
                        "\nWaiting 10 seconds for user and access keys to be created...\\n"
                    );
                    wait(10000);
                }
            }
        }
    }
}
```

```
        console.log(
            "Attempt to list S3 buckets with the new user (without permissions)...
\n"
);
// Use the credentials for the new user that you created.
var user_creds = {
    accessKeyId: myAccessKey,
    secretAccessKey: mySecretAccessKey,
};
const s3Client = new S3Client({
    credentials: user_creds,
    region: region,
});
await s3Client.send(new ListBucketsCommand({}));
} catch (err) {
    console.log(
        "Error. As expected the new user has no permissions to list buckets. ",
        err.stack
    );
    console.log(
        "\nCreating policy to allow the new user to list all buckets, and to
assume an STS role...\n"
);
    const myManagedPolicy = {
        Version: "2012-10-17",
        Statement: [
            {
                Effect: "Allow",
                Action: ["s3>ListAllMyBuckets", "sts:AssumeRole"],
                Resource: "*",
            },
        ],
    };
    const policy_params = {
        PolicyDocument: JSON.stringify(myManagedPolicy),
        PolicyName: s3_policy_name, // Name of the new policy.
    };
    const data = await iamClient.send(
        new CreatePolicyCommand(policy_params)
    );
    console.log(
        "Success. Policy created that allows listing of all S3 buckets.\n" +
        "Policy ARN: " +
        data.Policy.Arn +
        "\n" +
        "Policy name: " +
        data.Policy.PolicyName +
        "\n"
    );
    var s3_policy_arn = data.Policy.Arn;

    try {
        console.log(
            "\nCreating a role with a trust policy that lets the user assume the
role....\n"
        );
        const role_json = {
            Version: "2012-10-17",
            Statement: [
                {
                    Effect: "Allow",
                    Principal: {
                        AWS: user_arn, // The ARN of the user.
                    },
                },
            ],
        };
    }
}
```

```
        Action: "sts:AssumeRole",
    },
],
];
const myJson = JSON.stringify(role_json);

const role_params = {
    AssumeRolePolicyDocument: myJson, // Trust relationship policy
document.
    Path: "/",
    RoleName: role_name // The name of the new role.
};
const data = await iamClient.send(new CreateRoleCommand(role_params));
console.log("Success. Role created. Role Arn: ", data.Role.Arn);
const role_arn = data.Role.Arn;
try {
    console.log(
        "\nAttaching to the role the policy with permissions to list all
buckets....\n"
    );
    const params = {
        PolicyArn: s3_policy_arn,
        RoleName: role_name,
    };
    await iamClient.send(new AttachRolePolicyCommand(params));
    console.log("Success. Policy attached successfully to role.");
    try {
        console.log(
            "\nCreate a policy that enables the user to assume the role ....
\n"
        );
        const myNewPolicy = {
            Version: "2012-10-17",
            Statement: [
                {
                    Effect: "Allow",
                    Action: ["sts:AssumeRole"],
                    Resource: role_arn,
                },
            ],
        };
        const policy_params = {
            PolicyDocument: JSON.stringify(myNewPolicy),
            PolicyName: assume_policy_name,
        };
        const data = await iamClient.send(
            new CreatePolicyCommand(policy_params)
        );
        console.log(
            "Success. Policy created. Policy ARN: " + data.Policy.Arn
        );
    }
    const assume_policy_arn = data.Policy.Arn;
    try {
        console.log("\nAttaching the policy to the user....\n");
        const attach_policy_to_user_params = {
            PolicyArn: assume_policy_arn,
            UserName: user_name,
        };
        await iamClient.send(
            new AttachUserPolicyCommand(attach_policy_to_user_params)
        );
        console.log(
            "\nWaiting 10 seconds for policy to be attached...\n"
        );
        wait(10000);
    }
}
```

```
        console.log(
            "Success. Policy attached to user " + user_name + "."
        );
        try {
            console.log(
                "\nAssume for the user the role with permission to list all
buckets....\n"
            );
            const assume_role_params = {
                RoleArn: role_arn, //ARN_OF_ROLE_TO_ASSUME
                RoleSessionName: "session1",
                DurationSeconds: 900,
            };
            // Create an AWS STS client with the credentials for the user.
Remember, the user has permissions to assume roles using AWS STS.
            const stsClientWithUsersCreds = new STSClient({
                credentials: user_creds,
                region: REGION,
            });

            const data = await stsClientWithUsersCreds.send(
                new AssumeRoleCommand(assume_role_params)
            );
            console.log(
                "Success assuming role. Access key id is " +
                data.Credentials.AccessKeyId +
                "\n" +
                "Secret access key is " +
                data.Credentials.SecretAccessKey
            );

            const newAccessKey = data.Credentials.AccessKeyId;
            const newSecretAccessKey = data.Credentials.SecretAccessKey;

            console.log(
                "\nWaiting 10 seconds for the user to assume the role with
permission to list all buckets...\n"
            );
            wait(10000);
            // Set the parameters for the temporary credentials. This grants
permission to list S3 buckets.
            var new_role_creds = {
                accessKeyId: newAccessKey,
                secretAccessKey: newSecretAccessKey,
                sessionToken: data.Credentials.SessionToken,
            };
            try {
                console.log(
                    "Listing the S3 buckets using the credentials of the
assumed role... \n"
                );
                // Create an S3 client with the temporary credentials.
                const s3ClientWithNewCreds = new S3Client({
                    credentials: new_role_creds,
                    region: REGION,
                });
                const data = await s3ClientWithNewCreds.send(
                    new ListBucketsCommand({})
                );
                console.log("Success. Your S3 buckets are:", data.Buckets);
                try {
                    console.log(
                        "Detaching s3 policy from user " + userName + " ... \n"
                    );
                    await iamClient.send(
                        new DetachUserPolicyCommand({
```

```
        PolicyArn: assume_policy_arn,
        UserName: userName,
    })
);
console.log("Success, S3 policy detached from user.");
try {
    console.log(
        "Detaching role policy from " + role_name + " ... \n"
    );
    await iamClient.send(
        new DetachRolePolicyCommand({
            PolicyArn: s3_policy_arn,
            RoleName: role_name,
        })
    );
    console.log(
        "Success, assume policy detached from role."
    );
    try {
        console.log("Deleting s3 policy ... \n");
        await iamClient.send(
            new DeletePolicyCommand({
                PolicyArn: s3_policy_arn,
            })
        );
        console.log("Success, S3 policy deleted.");
        try {
            console.log("Deleting assume role policy ... \n");
            await iamClient.send(
                new DeletePolicyCommand({
                    PolicyArn: assume_policy_arn,
                })
            );
            try {
                console.log("Deleting access keys ... \n");
                await iamClient.send(
                    new DeleteAccessKeyCommand({
                        UserName: userName,
                        AccessKeyId: myAccessKey,
                    })
                );
            }
            try {
                console.log(
                    "Deleting user " + user_name + " ... \n"
                );
                await iamClient.send(
                    new DeleteUserCommand({ UserName: userName })
                );
                console.log("Success, user deleted.");
                try {
                    console.log(
                        "Deleting role " + role_name + " ... \n"
                    );
                    await iamClient.send(
                        new DeleteRoleCommand({
                            RoleName: role_name,
                        })
                    );
                    console.log("Success, role deleted.");
                    return "Run successfully"; // For unit tests.
                } catch (err) {
                    console.log("Error deleting role .", err);
                }
            } catch (err) {
                console.log("Error deleting user.", err);
            }
        }
    }
}
```

```
        } catch (err) {
            console.log("Error deleting access keys.", err);
        }
    } catch (err) {
        console.log(
            "Error detaching assume role policy from user.",
            err
        );
    }
} catch (err) {
    console.log("Error deleting role.", err);
}
} catch (err) {
    console.log("Error deleting user.", err);
}
} catch (err) {
    console.log("Error detaching S3 policy from role.", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error listing S3 buckets.", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error assuming role.", err);
    process.exit(1);
}
} catch (err) {
    console.log(
        "Error adding permissions to user to assume role.",
        err
    );
    process.exit(1);
}
} catch (err) {
    console.log("Error assuming role.", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error creating policy. ", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error attaching policy to role.", err);
    process.exit(1);
}
}
} catch (err) {
    console.log("Error creating access keys. ", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error creating user. ", err);
}
};

run(userName, s3_policy_name, role_name, assume_policy_name);
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)

- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap IAM user actions.

```
suspend fun main(args: Array<String>) {  
  
    val usage = """  
Usage:  
    <username> <policyName> <roleName> <roleSessionName> <fileLocation>  
<bucketName>  
  
    Where:  
        username - The name of the IAM user to create.  
        policyName - The name of the policy to create.  
        roleName - The name of the role to create.  
        roleSessionName - The name of the session required for the assumeRole  
operation.  
        fileLocation - The file location to the JSON required to create the role  
(see Readme).  
        bucketName - The name of the Amazon S3 bucket from which objects are read.  
    """  
  
    if (args.size != 6) {  
        println(usage)  
        exitProcess(1)  
    }  
  
    val userName = args[0]  
    val policyName = args[1]  
    val roleName = args[2]  
    val roleSessionName = args[3]  
    val fileLocation = args[4]  
    val bucketName = args[5]  
  
    createUser(userName)  
    println("$userName was successfully created.")  
  
    val polArn = createPolicy(policyName)  
    println("The policy $polArn was successfully created.")  
  
    val roleArn = createRole(roleName, fileLocation)  
    println("$roleArn was successfully created.")
```

```
        attachRolePolicy(roleName, polArn)

        println("**** Wait for 1 MIN so the resource is available.")
        delay(60000)
        assumeGivenRole(roleArn, roleSessionName, bucketName)

        println("**** Getting ready to delete the AWS resources.")
        deleteRole(roleName, polArn)
        deleteUser(userName)
        println("This IAM Scenario has successfully completed.")
    }

    suspend fun createUser(usernameVal: String?): String? {

        val request = CreateUserRequest {
            userName = usernameVal
        }

        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.createUser(request)
            return response.user?.userName
        }
    }

    suspend fun createPolicy(policyNameVal: String?): String {

        val policyDocumentValue: String = "{" +
            "  \"Version\": \"2012-10-17\", " +
            "  \"Statement\": [ " +
            "    { " +
            "      \"Effect\": \"Allow\", " +
            "      \"Action\": [ " +
            "        \"s3:*\" " +
            "      ], " +
            "      \"Resource\": \"*\" " +
            "    } " +
            "  ] " +
            "}"

        val request = CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.createPolicy(request)
            return response.policy?.arn.toString()
        }
    }

    suspend fun createRole(rolenameVal: String?, fileLocation: String?): String? {

        val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

        val request = CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toString()
            description = "Created using the AWS SDK for Kotlin"
        }

        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.createRole(request)
            return response.role?.arn
        }
    }
}
```

```
suspend fun attachRolePolicy(roleNameVal: String, policyArnVal: String) {  
  
    val request = ListAttachedRolePoliciesRequest {  
        roleName = roleNameVal  
    }  
  
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        val response = iamClient.listAttachedRolePolicies(request)  
        val attachedPolicies = response.attachedPolicies  
  
        // Ensure that the policy is not attached to this role.  
        val checkStatus: Int  
        if (attachedPolicies != null) {  
            checkStatus = checkMyList(attachedPolicies, policyArnVal)  
            if (checkStatus == -1)  
                return  
        }  
  
        val policyRequest = AttachRolePolicyRequest {  
            roleName = roleNameVal  
            policyArn = policyArnVal  
        }  
        iamClient.attachRolePolicy(policyRequest)  
        println("Successfully attached policy $policyArnVal to role $roleNameVal")  
    }  
}  
  
fun checkMyList(attachedPolicies: List<AttachedPolicy>, policyArnVal: String): Int {  
  
    for (policy in attachedPolicies) {  
        val polArn = policy.policyArn.toString()  
  
        if (polArn.compareTo(policyArnVal) == 0) {  
            println("The policy is already attached to this role.")  
            return -1  
        }  
    }  
    return 0  
}  
  
suspend fun assumeGivenRole(roleArnVal: String?, roleSessionNameVal: String?,  
    bucketName: String) {  
  
    val stsClient = StsClient {  
        region = "us-east-1"  
    }  
  
    val roleRequest = AssumeRoleRequest {  
        roleArn = roleArnVal  
        roleSessionName = roleSessionNameVal  
    }  
  
    val roleResponse = stsClient.assumeRole(roleRequest)  
    val myCreds = roleResponse.credentials  
    val key = myCreds?.accessKeyId  
    val secKey = myCreds?.secretAccessKey  
    val secToken = myCreds?.sessionToken  
  
    val staticCredentials = StaticCredentialsProvider {  
        accessKeyId = key  
        secretAccessKey = secKey  
        sessionToken = secToken  
    }  
  
    // List all objects in an Amazon S3 bucket using the temp creds.
```

```
val s3 = S3Client {
    credentialsProvider = staticCredentials
    region = "us-east-1"
}

println("Created a S3Client using temp credentials.")
println("Listing objects in $bucketName")

val listObjects = ListObjectsRequest {
    bucket = bucketName
}

val response = s3.listObjects(listObjects)
response.contents?.forEach { myObject ->
    println("The name of the key is ${myObject.key}")
    println("The owner is ${myObject.owner}")
}
}

suspend fun deleteRole(roleNameVal: String, polArn: String) {

    val iam = IamClient { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest = DetachRolePolicyRequest {
        policyArn = polArn
        roleName = roleNameVal
    }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request = DeletePolicyRequest {
        policyArn = polArn
    }

    iam.deletePolicy(request)
    println("**** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest = DeleteRoleRequest {
        roleName = roleNameVal
    }

    iam.deleteRole(roleRequest)
    println("**** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient { region = "AWS_GLOBAL" }
    val request = DeleteUserRequest {
        userName = userNameVal
    }

    iam.deleteUser(request)
    println("**** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

PHP

## SDK for PHP

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IamService;

echo("-----\n");
print("Welcome to the Amazon IAM getting started demo using PHP!\n");
echo("-----\n");
$uuid = uniqid();
$service = new IamService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Principal\": \"{AWS\": \"{$user['Arn']}\"},
            \"Action\": \"sts:AssumeRole\"
        }
    ]
}";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
    $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": \"s3:ListBucket\",
            \"Resource\": \"arn:aws:s3:::<your-bucket-name>\"
        }
    ]
}";
```

```
        \\"Action\\": \\\"s3>ListAllMyBuckets\\",
        \\"Resource\\": \\\"arn:aws:s3:::*\\"]]
    }];
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
    $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
    $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
    \\"Version\\": \\\"2012-10-17\\",
    \\"Statement\\": [{{
        \\"Effect\\": \\\"Allow\\",
        \\"Action\\": \\\"sts:AssumeRole\\",
        \\"Resource\\": \\\"{$assumeRoleRole['Arn']}\\"
    }}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
    $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
    'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
    'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
    'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should now run!\\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'],
    $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\\n";
```

- For API details, see the following topics in *AWS SDK for PHP API Reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an IAM user and a role that grants permission to list Amazon S3 buckets. The user has rights only to assume the role. After assuming the role, use temporary credentials to list buckets for the account.

```
import json
import sys
import time
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError


def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
        print('.', end='')
        sys.stdout.flush()
    print()


def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
                         that has permissions to create users, roles, and policies
                         in the account.
    :return: The newly created user, user key, and role.
    """
    # Create a new IAM user
    user = iam_resource.create_user(UserName='test-user')

    # Create an access key for the user
    access_key = user.create_access_key_pair()

    # Create an IAM role
    role = iam_resource.create_role(RoleName='test-role',
                                    AssumeRolePolicyDocument='{"Version": "2012-10-17", "Statement": [{"Action": "sts:AssumeRole", "Effect": "Allow", "Principal": "*"}]}')

    # Create a policy that allows listing S3 buckets
    s3_list_policy = iam_resource.create_policy(PolicyName='S3ListBucket',
                                                PolicyDocument='{"Version": "2012-10-17", "Statement": [{"Action": "s3:ListBucket", "Effect": "Allow", "Resource": "*"}]}')

    # Attach the policy to the role
    role.attach_policy(PolicyArn=s3_list_policy['Policy']['Arn'])

    # Create an inline policy for the user that lets them assume the role
    user.create_inline_policy(PolicyName='UserAssumeRolePolicy',
                            PolicyDocument='{"Version": "2012-10-17", "Statement": [{"Action": "sts:AssumeRole", "Effect": "Allow", "Principal": role['Role']['Arn"}]}')

    return user, access_key, role
```

```
try:
    user = iam_resource.create_user(UserName=f'demo-user-{uuid4()}')
    print(f"Created user {user.name}.")
except ClientError as error:
    print(f"Couldn't create a user for the demo. Here's why: "
          f"{error.response['Error']['Message']}")
    raise

try:
    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")
except ClientError as error:
    print(f"Couldn't create access keys for user {user.name}. Here's why: "
          f"{error.response['Error']['Message']}")
    raise

print(f"Wait for user to be ready.", end='')
progress_bar(10)

try:
    role = iam_resource.create_role(
        RoleName=f'demo-role-{uuid4()}',
        AssumeRolePolicyDocument=json.dumps({
            'Version': '2012-10-17',
            'Statement': [{
                'Effect': 'Allow',
                'Principal': {'AWS': user.arn},
                'Action': 'sts:AssumeRole']]))
    print(f"Created role {role.name}.")
except ClientError as error:
    print(f"Couldn't create a role for the demo. Here's why: "
          f"{error.response['Error']['Message']}")
    raise

try:
    policy = iam_resource.create_policy(
        PolicyName=f'demo-policy-{uuid4()}',
        PolicyDocument=json.dumps({
            'Version': '2012-10-17',
            'Statement': [{
                'Effect': 'Allow',
                'Action': 's3>ListAllMyBuckets',
                'Resource': 'arn:aws:s3:::*'}]}))
    role.attach_policy(PolicyArn=policy.arn)
    print(f"Created policy {policy.policy_name} and attached it to the role.")
except ClientError as error:
    print(f"Couldn't create a policy and attach it to role {role.name}. Here's
why: "
          f"{error.response['Error']['Message']}")
    raise

try:
    user.create_policy(
        PolicyName=f'demo-user-policy-{uuid4()}',
        PolicyDocument=json.dumps({
            'Version': '2012-10-17',
            'Statement': [{
                'Effect': 'Allow',
                'Action': 'sts:AssumeRole',
                'Resource': role.arn]}}))
    print(f"Created an inline policy for {user.name} that lets the user assume
"
          f"the role.")
except ClientError as error:
    print(f"Couldn't create an inline policy for user {user.name}. Here's why:
"
          f"{error.response['Error']['Message']}")
    raise
```

```
        f"{{error.response['Error']['Message']}}")
        raise

    print("Give AWS time to propagate these new resources and connections.",
end='')
    progress_bar(10)

    return user, user_key, role


def show_access_denied_without_role(user_key):
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
                     have permission to list buckets in the account.
    """
    print(f"Try to list buckets without first assuming the role.")
    s3_denied_resource = boto3.resource(
        's3', aws_access_key_id=user_key.id, aws_secret_access_key=user_key.secret)
    try:
        for bucket in s3_denied_resource.buckets.all():
            print(bucket.name)
        raise RuntimeError("Expected to get AccessDenied error when listing
buckets!")
    except ClientError as error:
        if error.response['Error']['Code'] == 'AccessDenied':
            print("Attempt to list buckets with no permissions: AccessDenied.")
        else:
            raise

def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
    Assumes a role that grants permission to list the Amazon S3 buckets in the
    account.
    Uses the temporary credentials from the role to list the buckets that are owned
    by the assumed role's account.

    :param user_key: The access key of a user that has permission to assume the
role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                           grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
        'sts', aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret)
    try:
        response = sts_client.assume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name)
        temp_credentials = response['Credentials']
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:
        print(f"Couldn't assume role {assume_role_arn}. Here's why: "
              f"{{error.response['Error']['Message']}}")
        raise

    # Create an S3 resource that can access the account with the temporary
    # credentials.
    s3_resource = boto3.resource(
        's3',
        aws_access_key_id=temp_credentials['AccessKeyId'],
        aws_secret_access_key=temp_credentials['SecretAccessKey'],
        aws_session_token=temp_credentials['SessionToken'])
    print(f"Listing buckets for the assumed role's account:")
```

```
try:
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
except ClientError as error:
    print(f"Couldn't list buckets for the account. Here's why: "
          f"{error.response['Error']['Message']}")
    raise

def teardown(user, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print("Couldn't detach policy, delete policy, or delete role. Here's why: "
              f"{error.response['Error']['Message']}")
        raise

    try:
        for user_pol in user.policies.all():
            user_pol.delete()
            print("Deleted inline user policy.")
        for key in user.access_keys.all():
            key.delete()
            print("Deleted user's access key.")
        user.delete()
        print(f"Deleted {user.name}.")
    except ClientError as error:
        print("Couldn't delete user policy or delete user. Here's why: "
              f"{error.response['Error']['Message']}")

def usage_demo():
    """Drives the demonstration."""
    print('*'*88)
    print(f"Welcome to the IAM create user and assume role demo.")
    print('-'*88)
    iam_resource = boto3.resource('iam')
    user = None
    role = None
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn, 'AssumeRoleDemoSession')
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
            teardown(user, role)
            print("Thanks for watching!")

if __name__ == '__main__':
    usage_demo()
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-iam"
require "aws-sdk-s3"

# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_resource

  # @param iam_resource [Aws::IAM::Resource] An AWS IAM resource.
  def initialize(iam_resource)
    @iam_resource = iam_resource
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
    puts("Give AWS time to propagate resources...")
    sleep(duration)
  end

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_resource.create_user(user_name: user_name)
    puts("Created demo user named #{user.name}.")
    rescue Aws::Errors::ServiceError => e
      puts("Tried and failed to create demo user.")
      puts("\t#{e.code}: #{e.message}")
      puts("\nCan't continue the demo without a user!")
      raise
    else
      user
  end
end
```

```
end

# Creates an access key for a user.
#
# @param user [Aws::IAM::User] The user that owns the key.
# @return [Aws::AccessKeyPair] The newly created access key.
def create_access_key_pair(user)
    user_key = user.create_access_key_pair
    puts("Created access key pair for user.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't create access keys for user #{user.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    user_key
end

# Creates a role that can be assumed by a user.
#
# @param role_name [String] The name to give the role.
# @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
# @return [Aws::IAM::Role] The newly created role.
def create_role(role_name, user)
    role = @iam_resource.create_role(
        role_name: role_name,
        assume_role_policy_document: {
            Version: "2012-10-17",
            Statement: [
                {
                    Effect: "Allow",
                    Principal: {'AWS': user.arn},
                    Action: "sts:AssumeRole"
                }
            ].to_json)
    puts("Created role #{role.name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't create a role for the demo. Here's why: ")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    role
end

# Creates a policy that grants permission to list S3 buckets in the account, and
# then attaches the policy to a role.
#
# @param policy_name [String] The name to give the policy.
# @param role [Aws::IAM::Role] The role that the policy is attached to.
# @return [Aws::IAM::Policy] The newly created policy.
def create_and_attach_role_policy(policy_name, role)
    policy = @iam_resource.create_policy(
        policy_name: policy_name,
        policy_document: {
            Version: "2012-10-17",
            Statement: [
                {
                    Effect: "Allow",
                    Action: "s3>ListAllMyBuckets",
                    Resource: "arn:aws:s3:::/*"
                }
            ].to_json)
    role.attach_policy(policy_arn: policy.arn)
    puts("Created policy #{policy.policy_name} and attached it to role
#{role.name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't create a policy and attach it to role #{role.name}. Here's why:
")
```

```
    puts("\t#{e.code}: #{e.message}")
    raise
else
  policy
end

# Creates an inline policy for a user that lets the user assume a role.
#
# @param policy_name [String] The name to give the policy.
# @param user [Aws::IAM::User] The user that owns the policy.
# @param role [Aws::IAM::Role] The role that can be assumed.
# @return [Aws::IAM::UserPolicy] The newly created policy.
def create_user_policy(policy_name, user, role)
  policy = user.create_policy(
    policy_name: policy_name,
    policy_document: {
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Action: "sts:AssumeRole",
          Resource: role.arn
        }
      ].to_json
    })
  puts("Created an inline policy for #{user.name} that lets the user assume role
#{role.name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't create an inline policy for user #{user.name}. Here's why: ")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  policy
end

# Creates an Amazon S3 resource with specified credentials. This is separated
into a
# factory function so that it can be mocked for unit testing.
#
# @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
def create_s3_resource(credentials)
  Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
end

# Lists the S3 buckets for the account, using the specified Amazon S3 resource.
# Because the resource uses credentials with limited access, it may not be able
to
# list the S3 buckets.
#
# @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
def list_buckets(s3_resource)
  count = 10
  s3_resource.buckets.each do |bucket|
    puts "\t#{bucket.name}"
    count -= 1
    break if count.zero?
  end
rescue Aws::Errors::ServiceError => e
  if e.code == "AccessDenied"
    puts("Attempt to list buckets with no permissions: AccessDenied.")
  else
    puts("Couldn't list buckets for the account. Here's why: ")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end
```

```
# Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
# This is separated into a factory function so that it can be mocked for unit
testing.
#
# @param key_id [String] The ID of the access key used by the STS client.
# @param key_secret [String] The secret part of the access key used by the STS
client.
def create_sts_client(key_id, key_secret)
    Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
end

# Gets temporary credentials that can be used to assume a role.
#
# @param role_arn [String] The ARN of the role that is assumed when these
credentials
#                               are used.
# @param sts_client [AWS::STS::Client] An AWS STS client.
# @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
def assume_role(role_arn, sts_client)
    credentials = Aws::AssumeRoleCredentials.new(
        client: sts_client,
        role_arn: role_arn,
        role_session_name: "create-use-assume-role-scenario"
    )
    puts("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
end

# Deletes a role. If the role has policies attached, they are detached and
# deleted before the role is deleted.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_role(role)
    role.attached_policies.each do |policy|
        name = policy.policy_name
        policy.detach_role(role_name: role.name)
        policy.delete
        puts("Deleted policy #{name}.")
    end
    name = role.name
    role.delete
    puts("Deleted role #{name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't detach policies and delete role #{role.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
end

# Deletes a user. If the user has inline policies or access keys, they are
deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
    user.policies.each do |policy|
        name = policy.name
        policy.delete
        puts("Deleted user policy #{name}.")
    end
    user.access_keys.each do |key|
        key.delete
        puts("Deleted access key for user #{user.name}.")
    end
    name = user.name
end
```

```

        user.delete
        puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
    puts("-" * 88)
    puts("Welcome to the IAM create a user and assume a role demo!")
    puts("-" * 88)

    user = scenario.create_user("doc-example-user-#{Random.uuid}")
    user_key = scenario.create_access_key_pair(user)
    scenario.wait(10)
    role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
    scenario.create_and_attach_role_policy("doc-example-role-policy-#{Random.uuid}", role)
    scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user, role)
    scenario.wait(10)
    puts("Try to list buckets with credentials for a user who has no permissions.")
    puts("Expect AccessDenied from this call.")
    scenario.list_buckets(
        scenario.create_s3_resource(Aws::Credentials.new(user_key.id,
user_key.secret)))
    puts("Now, assume the role that grants permission.")
    temp_credentials = scenario.assume_role(
        role.arn, scenario.create_sts_client(user_key.id, user_key.secret))
    puts("Here are your buckets:")
    scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
    puts("Deleting role '#{role.name}' and attached policies.")
    scenario.delete_role(role)
    puts("Deleting user '#{user.name}', policies, and keys.")
    scenario.delete_user(user)

    puts("Thanks for watching!")
    puts("-" * 88)
rescue Aws::Errors::ServiceError => e
    puts("Something went wrong with the demo.")
    puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Resource.new)) if
$PROGRAM_NAME == __FILE__

```

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)

- [PutUserPolicy](#)

Rust

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{Client as iamClient, Credentials as iamCredentials};
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use aws_types::region::Region;
use std::borrow::Borrow;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document) = initialize_variables().await;

    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{}: {}", e);
    }

    Ok(())
}

async fn initialize_variables() -> (iamClient, String, String, String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-west-2"));

    let shared_config =
        aws_config::from_env().region(region_provider).load().await;
    let client = iamClient::new(&shared_config);
    let uuid = Uuid::new_v4().to_string();

    let list_all_buckets_policy_document = "{\n        \"Version\": \"2012-10-17\",\n        \"Statement\": [\n            {\n                \"Effect\": \"Allow\",\n                \"Action\": \"s3>ListAllMyBuckets\",\n                \"Resource\": \"arn:aws:s3:::*\"\n            }\n        ].to_string();
    let inline_policy_document = "{\n        \"Version\": \"2012-10-17\",\n        \"Statement\": [\n
```

```

        \\"Effect\\": \\\"Allow\\",
        \\"Action\\": \\\"sts:AssumeRole\\",
        \\"Resource\\": \\\"{}\\\"]"
    }"
    .to_string();

    (
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
}

async fn run_iam_operations(
    client: iamClient,
    uuid: String,
    list_all_buckets_policy_document: String,
    inline_policy_document: String,
) -> Result<(), iamError> {
    let user = iam_service::create_user(&client, &format!("{}{}", "iam_demo_user_", uuid)).await?;
    println!(
        "Created the user with the name: {}",
        user.user_name.as_ref().unwrap()
    );
    let key = iam_service::create_access_key(&client,
        user.user_name.as_ref().unwrap()).await?;

    let assume_role_policy_document = "{\n        \\\"Version\\\": \\\"2012-10-17\\\",\\n        \\\"Statement\\\": [\\n            {\n                \\\"Effect\\\": \\\"Allow\\",
                \\"Principal\\": \\\"AWS\\": \\\"{}\\\"",
                \\"Action\\": \\\"sts:AssumeRole\\"
            }
        ]"
    .to_string()
    .replace("{}", user.arn.as_ref().unwrap());

    let assume_role_role = iam_service::create_role(
        &client,
        &format!("{}{}", "iam_demo_role_", uuid),
        &assume_role_policy_document,
    )
    .await?;
    println!(
        "Created the role with the ARN: {}",
        assume_role_role.arn.as_ref().unwrap()
    );

    let list_all_buckets_policy = iam_service::create_policy(
        &client,
        &format!("{}{}", "iam_demo_policy_", uuid),
        &list_all_buckets_policy_document,
    )
    .await?;
    println!(
        "Created policy: {}",
        list_all_buckets_policy.policy_name.as_ref().unwrap()
    );

    let attach_role_policy_result =
        iam_service::attach_role_policy(&client, &assume_role_role,
        &list_all_buckets_policy)
        .await?;
}

```

```
    println!(
        "Attached the policy to the role: {:?}",
        attach_role_policy_result
    );

    let inline_policy_name = format!("{}{}", "iam_demo_inline_policy_", uuid);
    let inline_policy_document =
        inline_policy_document.replace("{}",
assume_role_role.arn.as_ref().unwrap());
    iam_service::create_user_policy(&client, &user, &inline_policy_name,
&inline_policy_document)
        .await?;
    println!("Created inline policy.");

    //First, fail to list the buckets with the user.
    let creds = iamCredentials::from_keys(
        key.access_key_id.as_ref().unwrap(),
        key.secret_access_key.as_ref().unwrap(),
        None,
    );
    let fail_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    println!("Fail config: {:?}", fail_config);
    let fail_client: s3Client = s3Client::new(&fail_config);
    match fail_client.list_buckets().send().await {
        Ok(e) => {
            println!("This should not run. {:?}", e);
        }
        Err(e) => {
            println!("Successfully failed with error: {:?}", e)
        }
    }
}

let sts_config = aws_config::from_env()
    .credentials_provider(creds.clone())
    .load()
    .await;
let sts_client: stsClient = stsClient::new(&sts_config);
sleep(Duration::from_secs(10)).await;
let assumed_role = sts_client
    .assume_role()
    .role_arn(assume_role_role.arn.as_ref().unwrap())
    .role_session_name(&format!("{}{}", "iam_demo_assumerole_session_", uuid))
    .send()
    .await;
println!("Assumed role: {:?}", assumed_role);
sleep(Duration::from_secs(10)).await;

let assumed_credentials = iamCredentials::from_keys(
    assumed_role
        .as_ref()
        .unwrap()
        .credentials
        .as_ref()
        .unwrap()
        .access_key_id
        .as_ref()
        .unwrap(),
    assumed_role
        .as_ref()
        .unwrap()
        .credentials
        .as_ref()
        .unwrap()
)
```

```

        .secret_access_key
        .as_ref()
        .unwrap(),
    assumed_role
        .as_ref()
        .unwrap()
        .credentials
        .as_ref()
        .unwrap()
        .session_token
        .borrow()
        .clone(),
    );
}

let succeed_config = aws_config::from_env()
    .credentials_provider(assumed_credentials)
    .load()
    .await;
println!("succeed config: {:?}", succeed_config);
let succeed_client: s3Client = s3Client::new(&succeed_config);
sleep(Duration::from_secs(10)).await;
match succeed_client.list_buckets().send().await {
    Ok(_) => {
        println!("This should now run successfully.")
    }
    Err(e) => {
        println!("This should not run. {:?}", e);
        panic!()
    }
}
//Clean up.
iam_service::detach_role_policy(
    &client,
    assume_role_role.role_name.as_ref().unwrap(),
    list_all_buckets_policy.arn.as_ref().unwrap(),
)
.await?;
iam_service::delete_policy(&client, list_all_buckets_policy).await?;
iam_service::delete_role(&client, &assume_role_role).await?;
println!(
    "Deleted role {}",
    assume_role_role.role_name.as_ref().unwrap()
);
iam_service::delete_access_key(&client, &user, &key).await?;
println!("Deleted key for {}", key.user_name.as_ref().unwrap());
iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
println!("Deleted inline user policy: {}", inline_policy_name);
iam_service::delete_user(&client, &user).await?;
println!("Deleted user {}", user.user_name.as_ref().unwrap());

Ok(())
}

```

- For API details, see the following topics in *AWS SDK for Rust API reference*.

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)

- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## Create read-only and read-write IAM users using an AWS SDK

The following code example shows how to:

- Create two IAM users.
- Attach a policy that lets one of the users get and put objects in an Amazon Simple Storage Service (Amazon S3) bucket.
- Attach a policy that lets the other user get objects from the bucket.
- Obtain differing permissions to the bucket based on user credentials.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap IAM user actions.

```
import logging
import time

import boto3
from botocore.exceptions import ClientError

import access_key_wrapper
import policy_wrapper

logger = logging.getLogger(__name__)
iam = boto3.resource('iam')

def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user

def update_user(user_name, new_user_name):
    """
```

```
Updates a user's name.

:param user_name: The current name of the user to update.
:param new_user_name: The new name to assign to the user.
:return: The updated user.
"""
try:
    user = iam.User(user_name)
    user.update(NewUserName=new_user_name)
    logger.info("Renamed %s to %s.", user_name, new_user_name)
except ClientError:
    logger.exception("Couldn't update name for user %s.", user_name)
    raise
return user

def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
try:
    users = list(iam.users.all())
    logger.info("Got %s users.", len(users))
except ClientError:
    logger.exception("Couldn't get users.")
    raise
else:
    return users

def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
try:
    iam.User(user_name).delete()
    logger.info("Deleted user %s.", user_name)
except ClientError:
    logger.exception("Couldn't delete user %s.", user_name)
    raise

def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
try:
    iam.User(user_name).attach_policy(PolicyArn=policy_arn)
    logger.info("Attached policy %s to user %s.", policy_arn, user_name)
except ClientError:
    logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
user_name)
    raise

def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """

```

```
try:
    iam.User(user_name).detach_policy(PolicyArn=policy_arn)
    logger.info("Detached policy %s from user %s.", policy_arn, user_name)
except ClientError:
    logger.exception(
        "Couldn't detach policy %s from user %s.", policy_arn, user_name)
    raise
```

Create functions that wrap IAM policy actions.

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource('iam')

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy
                        applies to. This ARN can contain wildcards, such as
                        'arn:aws:s3:::my-bucket/*' to allow actions on all objects
                        in the bucket named 'my-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": actions,
                "Resource": resource_arn
            }
        ]
    }
    try:
        policy = iam.create_policy(
            PolicyName=name, Description=description,
            PolicyDocument=json.dumps(policy_doc))
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy

def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
```

```
    iam.Policy(policy_arn).delete()
    logger.info("Deleted policy %s.", policy_arn)
except ClientError:
    logger.exception("Couldn't delete policy %s.", policy_arn)
    raise
```

Create functions that wrap IAM access key actions.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource('iam')

def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name, key_pair.id)
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair

def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info(
            "Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

Use the wrapper functions to create users with differing policies and use their credentials to access an Amazon S3 bucket.

```
def usage_demo():
    """
    Shows how to manage users, keys, and policies.
    This demonstration creates two users: one user who can put and get objects in
    an Amazon S3 bucket, and another user who can only get objects from the bucket.
```

```
The demo then shows how the users can perform only the actions they are
permitted
to perform.
"""
logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
print('*'*88)
print("Welcome to the AWS Identity and Account Management user demo.")
print('*'*88)
print("Users can have policies and roles attached to grant them specific "
      "permissions.")
s3 = boto3.resource('s3')
bucket = s3.create_bucket(
    Bucket=f'demo-iam-bucket-{time.time_ns()}',
    CreateBucketConfiguration={
        'LocationConstraint': s3.meta.client.meta.region_name
    }
)
print(f"Created an Amazon S3 bucket named {bucket.name}.")
user_read_writer = create_user('demo-iam-read-writer')
user_reader = create_user('demo-iam-reader')
print(f"Created two IAM users: {user_read_writer.name} and {user_reader.name}")
update_user(user_read_writer.name, 'demo-iam-creator')
update_user(user_reader.name, 'demo-iam-getter')
users = list_users()
user_read_writer = next(user for user in users if user.user_id ==
user_read_writer.user_id)
user_reader = next(user for user in users if user.user_id ==
user_reader.user_id)
print(f"Changed the names of the users to {user_read_writer.name} "
      "and {user_reader.name}.")

read_write_policy = policy_wrapper.create_policy(
    'demo-iam-read-write-policy',
    'Grants rights to create and get an object in the demo bucket.',
    ['s3:PutObject', 's3:GetObject'],
    f'arn:aws:s3::{bucket.name}/*')
print(f"Created policy {read_write_policy.policy_name} with ARN:
{read_write_policy.arn}")
print(read_write_policy.description)
read_policy = policy_wrapper.create_policy(
    'demo-iam-read-policy',
    'Grants rights to get an object from the demo bucket.',
    's3:GetObject',
    f'arn:aws:s3::{bucket.name}/*')
print(f"Created policy {read_policy.policy_name} with ARN: {read_policy.arn}")
print(read_policy.description)
attach_policy(user_read_writer.name, read_write_policy.arn)
print(f"Attached {read_write_policy.policy_name} to {user_read_writer.name}.")
attach_policy(user_reader.name, read_policy.arn)
print(f"Attached {read_policy.policy_name} to {user_reader.name}.")

user_read_writer_key = access_key_wrapper.create_key(user_read_writer.name)
print(f"Created access key pair for {user_read_writer.name}.")
user_reader_key = access_key_wrapper.create_key(user_reader.name)
print(f"Created access key pair for {user_reader.name}.")

s3_read_writer_resource = boto3.resource(
    's3',
    aws_access_key_id=user_read_writer_key.id,
    aws_secret_access_key=user_read_writer_key.secret)
demo_object_key = f'object-{time.time_ns()}'
demo_object = None
while demo_object is None:
    try:
        demo_object = s3_read_writer_resource.Bucket(bucket.name).put_object(
            Key=demo_object_key, Body=b'AWS IAM demo object content!')
```

```
        except ClientError as error:
            if error.response['Error']['Code'] == 'InvalidAccessKeyId':
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise
        print(f"Put {demo_object_key} into {bucket.name} using "
              f"[{user_read_writer.name}]'s credentials.")

    read_writer_object = s3_read_writer_resource.Bucket(
        bucket.name).Object(demo_object_key)
    read_writer_content = read_writer_object.get()['Body'].read()
    print(f"Got object {read_writer_object.key} using read-writer user's
credentials.")
    print(f"Object content: {read_writer_content}")

    s3_reader_resource = boto3.resource(
        's3',
        aws_access_key_id=user_reader_key.id,
        aws_secret_access_key=user_reader_key.secret)
    demo_content = None
    while demo_content is None:
        try:
            demo_object =
            s3_reader_resource.Bucket(bucket.name).Object(demo_object_key)
            demo_content = demo_object.get()['Body'].read()
            print(f"Got object {demo_object.key} using reader user's credentials.")
            print(f"Object content: {demo_content}")
        except ClientError as error:
            if error.response['Error']['Code'] == 'InvalidAccessKeyId':
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise

    try:
        demo_object.delete()
    except ClientError as error:
        if error.response['Error']['Code'] == 'AccessDenied':
            print('*'*88)
            print("Tried to delete the object using the reader user's credentials.
"
                  "Got expected AccessDenied error because the reader is not "
                  "allowed to delete objects.")
            print('*'*88)

    access_key_wrapper.delete_key(user_reader.name, user_reader_key.id)
    detach_policy(user_reader.name, read_policy.arn)
    policy_wrapper.delete_policy(read_policy.arn)
    delete_user(user_reader.name)
    print(f"Deleted keys, detached and deleted policy, and deleted
{user_reader.name}.")

    access_key_wrapper.delete_key(user_read_writer.name, user_read_writer_key.id)
    detach_policy(user_read_writer.name, read_write_policy.arn)
    policy_wrapper.delete_policy(read_write_policy.arn)
    delete_user(user_read_writer.name)
    print(f"Deleted keys, detached and deleted policy, and deleted
{user_read_writer.name}.")

    bucket.objects.delete()
    bucket.delete()
    print(f"Emptied and deleted {bucket.name}.")
    print("Thanks for watching!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [AttachUserPolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteUser](#)
  - [DetachUserPolicy](#)
  - [ListUsers](#)
  - [UpdateUser](#)

## Manage IAM access keys using an AWS SDK

The following code example shows how to:

- Create and list access keys.
- Find out when and how an access key was last used.
- Update and delete access keys.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap IAM access key actions.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource('iam')

def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys

def create_key(user_name):
    """
```

```
Creates an access key for the specified user. Each user can have a
maximum of two keys.

:param user_name: The name of the user.
:return: The created access key.
"""
try:
    key_pair = iam.User(user_name).create_access_key_pair()
    logger.info(
        "Created access key pair for %s. Key ID is %s.",
        key_pair.user_name, key_pair.id)
except ClientError:
    logger.exception("Couldn't create access key pair for %s.", user_name)
    raise
else:
    return key_pair

def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response['AccessKeyLastUsed'].get('LastUsedDate', None)
        last_service = response['AccessKeyLastUsed'].get('ServiceName', None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.", key_id,
            response['UserName'], last_used_date, last_service)
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response

def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
    deactivated.
    """
    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", 'Activated' if activate else 'Deactivated',
key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", 'Activate' if activate else 'Deactivate',
key_id)
        raise

def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    """

```

```
:param key_id: The ID of the key to delete.
"""

try:
    key = iam.AccessKey(user_name, key_id)
    key.delete()
    logger.info(
        "Deleted access key %s for %s.", key.id, key.user_name)
except ClientError:
    logger.exception("Couldn't delete key %s for %s", key_id, user_name)
    raise
```

Use the wrapper functions to perform access key actions for the current user.

```
def usage_demo():
    """Shows how to create and manage access keys."""
    def print_keys():
        """Gets and prints the current keys for a user."""
        current_keys = list_keys(current_user_name)
        print("The current user's keys are now:")
        print(*[f"{key.id}: {key.status}" for key in current_keys], sep='\n')

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    print('*'*88)
    print("Welcome to the AWS Identity and Account Management access key demo.")
    print('*'*88)
    current_user_name = iam.CurrentUser().user_name
    print(f"This demo creates an access key for the current user "
          f"{{current_user_name}}, manipulates the key in a few ways, and then "
          f"deletes it.")
    all_keys = list_keys(current_user_name)
    if len(all_keys) == 2:
        print("The current user already has the maximum of 2 access keys. To run "
              "this demo, either delete one of the access keys or use a user "
              "that has only 1 access key.")
    else:
        new_key = create_key(current_user_name)
        print(f"Created a new key with id {new_key.id} and secret "
              f"{new_key.secret}.")
        print_keys()
        existing_key = next(key for key in all_keys if key != new_key)
        last_use = get_last_use(existing_key.id)['AccessKeyLastUsed']
        print(f"Key {all_keys[0].id} was last used to access "
              f"{{last_use['ServiceName']} }"
              f"on {{last_use['LastUsedDate']} }")
        update_key(current_user_name, new_key.id, False)
        print(f"Key {new_key.id} is now deactivated.")
        print_keys()
        delete_key(current_user_name, new_key.id)
        print_keys()
        print("Thanks for watching!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateAccessKey](#)
  - [DeleteAccessKey](#)
  - [GetAccessKeyLastUsed](#)
  - [ListAccessKeys](#)
  - [UpdateAccessKey](#)

## Manage IAM policies using an AWS SDK

The following code example shows how to:

- Create and list policies.
- Create and get policy versions.
- Roll back a policy to a previous version.
- Delete policies.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap IAM policy actions.

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource('iam')

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
        form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy
        applies to. This ARN can contain wildcards, such as
        'arn:aws:s3:::my-bucket/*' to allow actions on all objects
        in the bucket named 'my-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": actions,
                "Resource": resource_arn
            }
        ]
    }
    try:
        policy = iam.create_policy(
            PolicyName=name, Description=description,
            PolicyDocument=json.dumps(policy_doc))
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
```

```
        raise
    else:
        return policy

def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are
                  returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies

def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                          version for the policy. Otherwise, the default
                          is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Action': actions,
                'Resource': resource_arn
            }
        ]
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default)
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id, policy_version.arn)
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version

def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
```

```

        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get('Statement', None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
policy_arn)
        raise
    else:
        return policy_statement

def rollback_policy_version(policy_arn):
"""
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

:param policy_arn: The ARN of the policy to roll back.
:return: The default version of the policy after the rollback.
"""
try:
    policy_versions = sorted(
        iam.Policy(policy_arn).versions.all(),
        key=operator.attrgetter('create_date'))
    logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
except ClientError:
    logger.exception("Couldn't get versions for %s.", policy_arn)
    raise

default_version = None
rollback_version = None
try:
    while default_version is None:
        ver = policy_versions.pop()
        if ver.is_default_version:
            default_version = ver
    rollback_version = policy_versions.pop()
    rollback_version.set_as_default()
    logger.info("Set %s as the default version.", rollback_version.version_id)
    default_version.delete()
    logger.info("Deleted original default version %s.",
default_version.version_id)
except IndexError:
    if default_version is None:
        logger.warning("No default version found for %s.", policy_arn)
    elif rollback_version is None:
        logger.warning(
            "Default version %s found for %s, but no previous version exists,
so "
            "nothing to roll back to.", default_version.version_id, policy_arn)
except ClientError:
    logger.exception("Couldn't roll back version for %s.", policy_arn)
    raise
else:
    return rollback_version

def delete_policy(policy_arn):
"""
    Deletes a policy.

:param policy_arn: The ARN of the policy to delete.

```

```
"""
try:
    iam.Policy(policy_arn).delete()
    logger.info("Deleted policy %s.", policy_arn)
except ClientError:
    logger.exception("Couldn't delete policy %s.", policy_arn)
    raise
```

Use the wrapper functions to create policies, update versions, and get information about them.

```
def usage_demo():
    """Shows how to use the policy functions."""
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    print('*'*88)
    print("Welcome to the AWS Identity and Account Management policy demo.")
    print('*'*88)
    print("Policies let you define sets of permissions that can be attached to "
          "other IAM resources, like users and roles.")
    bucket_arn = f'arn:aws:s3:::made-up-bucket-name'
    policy = create_policy(
        'demo-iam-policy', 'Policy for IAM demonstration.',
        ['s3>ListObjects'], bucket_arn)
    print(f"Created policy {policy.policy_name}.")
    policies = list_policies('Local')
    print(f"Your account has {len(policies)} managed policies:")
    print(*[pol.policy_name for pol in policies], sep=', ')
    time.sleep(1)
    policy_version = create_policy_version(
        policy.arn, ['s3:PutObject'], bucket_arn, True)
    print(f"Added policy version {policy_version.version_id} to policy "
          f"{policy.policy_name}.")
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is:")
    pprint.pprint(default_statement)
    rollback_version = rollback_policy_version(policy.arn)
    print(f"Rolled back to version {rollback_version.version_id} for "
          f"{policy.policy_name}.")
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is now:")
    pprint.pprint(default_statement)
    delete_policy(policy.arn)
    print(f"Deleted policy {policy.policy_name}.")
    print("Thanks for watching!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreatePolicy](#)
  - [CreatePolicyVersion](#)
  - [DeletePolicy](#)
  - [DeletePolicyVersion](#)
  - [GetPolicyVersion](#)
  - [ListPolicies](#)
  - [ListPolicyVersions](#)
  - [SetDefaultPolicyVersion](#)

## Manage IAM roles using an AWS SDK

The following code example shows how to:

- Create an IAM role.
- Attach and detach policies for a role.
- Delete a role.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap IAM role actions.

```
import json
import logging
import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource('iam')

def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Principal': {'Service': service},
                'Action': 'sts:AssumeRole'
            } for service in allowed_services
        ]
    }

    try:
        role = iam.create_role(
            RoleName=role_name,
            AssumeRolePolicyDocument=json.dumps(trust_policy))
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role

def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn, role_name)
```

```

        except ClientError:
            logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
role_name)
            raise

def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name)
        raise

def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise

```

Use the wrapper functions to create a role, then attach and detach a policy.

```

def usage_demo():
    """Shows how to use the role functions."""
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    print('-'*88)
    print("Welcome to the AWS Identity and Account Management role demo.")
    print('-'*88)
    print("Roles let you define sets of permissions and can be assumed by "
          "other entities, like users and services.")
    print("The first 10 roles currently in your account are:")
    roles = list_roles(10)
    print(f"The inline policies for role {roles[0].name} are:")
    list_policies(roles[0].name)
    role = create_role(
        'demo-iam-role',
        ['lambda.amazonaws.com', 'batchoperations.s3.amazonaws.com'])
    print(f"Created role {role.name}, with trust policy:")
    pprint.pprint(role.assume_role_policy_document)
    policy_arn = 'arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess'
    attach_policy(role.name, policy_arn)
    print(f"Attached policy {policy_arn} to {role.name}.")
    print(f"Policies attached to role {role.name} are:")
    list_attached_policies(role.name)
    detach_policy(role.name, policy_arn)
    print(f"Detached policy {policy_arn} from {role.name}.")
    delete_role(role.name)
    print(f"Deleted {role.name}.")
    print("Thanks for watching!")

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [AttachRolePolicy](#)
  - [CreateRole](#)
  - [DeleteRole](#)
  - [DetachRolePolicy](#)

## Manage your IAM account using an AWS SDK

The following code example shows how to:

- Get and update the alias for the account.
- Generate a report of users and their credentials.
- Get a summary of account usage.
- Get details about all users, groups, roles, and policies in your account, including their relationships to each other.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap IAM account actions.

```
import logging
import pprint
import sys
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource('iam')

def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
    alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response['AccountAliases']
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ', '.join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response['AccountAliases']

def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of the
    account name when assuming roles or using access keys.
    """
    # Create the alias
    response = iam.create_account_alias(AccountAlias=alias)
    logger.info("Created account alias: %s.", alias)

    # Verify the alias was created
    aliases = list_aliases()
    if alias in aliases:
        logger.info("Alias %s successfully created.", alias)
    else:
        logger.error("Alias %s failed to create.", alias)
```

```
account ID in the sign-in URL. An account can have only one alias. When a new
alias is created, it replaces any existing alias.

:param alias: The alias to assign to the account.
"""

try:
    iam.create_account_alias(AccountAlias=alias)
    logger.info("Created an alias '%s' for your account.", alias)
except ClientError:
    logger.exception("Couldn't create alias '%s' for your account.", alias)
    raise

def delete_alias(alias):
"""
Removes the alias from the current account.

:param alias: The alias to remove.
"""
try:
    iam.meta.client.delete_account_alias(AccountAlias=alias)
    logger.info("Removed alias '%s' from your account.", alias)
except ClientError:
    logger.exception("Couldn't remove alias '%s' from your account.", alias)
    raise

def generate_credential_report():
"""
Starts generation of a credentials report about the current account. After
calling this function to generate the report, call get_credential_report
to get the latest report. A new report can be generated a minimum of four hours
after the last one was generated.
"""
try:
    response = iam.meta.client.generate_credential_report()
    logger.info("Generating credentials report for your account. "
               "Current state is %s.", response['State'])
except ClientError:
    logger.exception("Couldn't generate a credentials report for your
account.")
    raise
else:
    return response

def get_credential_report():
"""
Gets the most recently generated credentials report about the current account.

:return: The credentials report.
"""
try:
    response = iam.meta.client.get_credential_report()
    logger.debug(response['Content'])
except ClientError:
    logger.exception("Couldn't get credentials report.")
    raise
else:
    return response['Content']

def get_summary():
"""
Gets a summary of account usage.

:return: The summary of account usage.
"""
try:
```

```

        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map

def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report, such
                           as users or roles. When not specified, all resources
                           are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details

```

Call wrapper functions to change the account alias and to get reports about the account.

```

def usage_demo():
    """
    Shows how to use the account functions.
    """
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    print('*'*88)
    print("Welcome to the AWS Identity and Account Management account demo.")
    print('*'*88)
    print("Setting an account alias lets you use the alias in your sign-in URL "
          "instead of your account number.")
    old_aliases = list_aliases()
    if len(old_aliases) > 0:
        print(f"Your account currently uses '{old_aliases[0]}' as its alias.")
    else:
        print("Your account currently has no alias.")
    for index in range(1, 3):
        new_alias = f'alias-{index}-{time.time_ns()}''
        print(f"Setting your account alias to {new_alias}")
        create_alias(new_alias)
    current_aliases = list_aliases()
    print(f"Your account alias is now {current_aliases}.")
    delete_alias(current_aliases[0])
    print(f"Your account now has no alias.")
    if len(old_aliases) > 0:
        print(f"Restoring your original alias back to {old_aliases[0]}...")
        create_alias(old_aliases[0])

    print('*'*88)
    print("You can get various reports about your account.")
    print("Let's generate a credentials report...")
    report_state = None
    while report_state != 'COMPLETE':
        cred_report_response = generate_credential_report()
        old_report_state = report_state
        report_state = cred_report_response['State']
        if report_state != old_report_state:

```

```

        print(report_state, sep='')
    else:
        print('.', sep='')
    sys.stdout.flush()
    time.sleep(1)
print()
cred_report = get_credential_report()
col_count = 3
print(f"Got credentials report. Showing only the first {col_count} columns.")
cred_lines = [line.split(',')[:col_count] for line
             in cred_report.decode('utf-8').split('\n')]
col_width = max([len(item) for line in cred_lines for item in line]) + 2
for line in cred_report.decode('utf-8').split('\n'):
    print(''.join(element.ljust(col_width)
                  for element in line.split(',')[:col_count]))

print('*'*88)
print("Let's get an account summary.")
summary = get_summary()
print("Here's your summary:")
pprint.pprint(summary)

print('*'*88)
print("Let's get authorization details!")
details = get_authorization_details([])
see_details = input("These are pretty long, do you want to see them (y/n)? ")
if see_details.lower() == 'y':
    pprint.pprint(details)

print('*'*88)
pw_policy_created = None
see_pw_policy = input("Want to see the password policy for the account (y/n)? ")
if see_pw_policy.lower() == 'y':
    while True:
        if print_password_policy():
            break
        else:
            answer = input("Do you want to create a default password policy (y/n)? ")
            if answer.lower() == 'y':
                pw_policy_created = iam.create_account_password_policy()
            else:
                break
    if pw_policy_created is not None:
        answer = input("Do you want to delete the password policy (y/n)? ")
        if answer.lower() == 'y':
            pw_policy_created.delete()
            print("Password policy deleted.")

print("The SAML providers for your account are:")
list_saml_providers(10)

print('*'*88)
print("Thanks for watching.")

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateAccountAlias](#)
  - [DeleteAccountAlias](#)
  - [GenerateCredentialReport](#)
  - [GetAccountAuthorizationDetails](#)
  - [GetAccountSummary](#)

- [GetCredentialReport](#)
- [ListAccountAliases](#)

## Roll back an IAM policy version using an AWS SDK

The following code example shows how to:

- Get the list of policy versions in order by date.
- Find the default policy version.
- Make the previous policy version the default.
- Delete the old default version.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter('create_date'))
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
        logger.info("Set %s as the default version.", rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
                   default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
```

```
        logger.warning(
            "Default version %s found for %s, but no previous version exists",
            so "
                "nothing to roll back to.", default_version.version_id, policy_arn)
        except ClientError:
            logger.exception("Couldn't roll back version for %s.", policy_arn)
            raise
    else:
        return rollback_version
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [DeletePolicyVersion](#)
  - [ListPolicyVersions](#)
  - [SetDefaultPolicyVersion](#)

## Cross-service examples for IAM using AWS SDKs

The following code examples show how to use AWS Identity and Access Management with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Create a long-lived Amazon EMR cluster and run several steps using an AWS SDK \(p. 1098\)](#)
- [Create a short-lived Amazon EMR cluster and run a step using an AWS SDK \(p. 1099\)](#)

## Create a long-lived Amazon EMR cluster and run several steps using an AWS SDK

The following code example shows how to create a long-lived Amazon EMR cluster and run several steps.

### Python

#### SDK for Python (Boto3)

Create a long-lived Amazon EMR cluster that uses Apache Spark to query historical Amazon review data from the [Amazon Customer Reviews Dataset](#). Run a job that gets data for top-rated products in specific categories that contain keywords in their product titles. Job results are written to an Amazon Simple Storage Service (Amazon S3) bucket.

- Create an Amazon S3 bucket and upload a job script.
- Create AWS Identity and Access Management (IAM) roles.
- Create Amazon Elastic Compute Cloud (Amazon EC2) security groups.
- Create a long-lived cluster and run several job steps.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon EC2
- Amazon EMR
- IAM
- Amazon S3

## Create a short-lived Amazon EMR cluster and run a step using an AWS SDK

The following code example shows how to create a short-lived Amazon EMR cluster that runs a step and automatically terminates after the step completes.

Python

### SDK for Python (Boto3)

Create a short-lived Amazon EMR cluster that estimates the value of pi using Apache Spark to parallelize a large number of calculations. The job writes output to Amazon EMR logs and to an Amazon Simple Storage Service (Amazon S3) bucket. The cluster terminates itself after completing the job.

- Create an Amazon S3 bucket and upload a job script.
- Create AWS Identity and Access Management (IAM) roles.
- Create Amazon Elastic Compute Cloud (Amazon EC2) security groups.
- Create a short-lived cluster and run a single job step.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon EC2
- Amazon EMR
- IAM
- Amazon S3

## Code examples for AWS IoT using AWS SDKs

The following code examples show how to use AWS IoT with an AWS software development kit (SDK).

### Code examples

- [Actions for AWS IoT using AWS SDKs \(p. 1099\)](#)
  - [Get information about an endpoint using an AWS SDK \(p. 1099\)](#)
  - [List your AWS IoT things using an AWS SDK \(p. 1100\)](#)

## Actions for AWS IoT using AWS SDKs

The following code examples show how to use AWS IoT with AWS SDKs. Each example calls an individual service function.

### Examples

- [Get information about an endpoint using an AWS SDK \(p. 1099\)](#)
- [List your AWS IoT things using an AWS SDK \(p. 1100\)](#)

## Get information about an endpoint using an AWS SDK

The following code example shows how to get AWS IoT endpoint information.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_address(client: &Client, endpoint_type: &str) -> Result<(), Error> {
    let resp = client
        .describe_endpoint()
        .endpoint_type(endpoint_type)
        .send()
        .await?;

    println!("Endpoint address: {}", resp.endpoint_address.unwrap());

    println!();
    Ok(())
}
```

- For API details, see [DescribeEndpoint](#) in *AWS SDK for Rust API reference*.

## List your AWS IoT things using an AWS SDK

The following code example shows how to list your AWS IoT things.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_things(client: &Client) -> Result<(), Error> {
    let resp = client.list_things().send().await?;

    println!("Things:");

    for thing in resp.things.unwrap() {
        println!(
            "  Name: {}",
            thing.thing_name.as_deref().unwrap_or_default()
        );
        println!(
            "  Type: {}",
            thing.thing_type_name.as_deref().unwrap_or_default()
        );
    }
}
```

```
    println!(
        "  ARN: {}",
        thing.thing_arn.as_deref().unwrap_or_default()
    );
    println!();
}
println!();
Ok(())
}
```

- For API details, see [ListThings](#) in *AWS SDK for Rust API reference*.

## Code examples for AWS KMS using AWS SDKs

The following code examples show how to use AWS Key Management Service with an AWS software development kit (SDK).

### Code examples

- [Actions for AWS KMS using AWS SDKs \(p. 1102\)](#)
  - Create a grant for an AWS KMS key using an AWS SDK (p. 1102)
  - Create an AWS KMS key using an AWS SDK (p. 1105)
  - Generate a random byte string using an AWS SDK (p. 1109)
  - Create an alias for an AWS KMS key using an AWS SDK (p. 1110)
  - Decrypt ciphertext with an AWS KMS key using an AWS SDK (p. 1112)
  - Delete an AWS KMS alias using an AWS SDK (p. 1116)
  - Describe an AWS KMS key using an AWS SDK (p. 1117)
  - Disable an AWS KMS key using an AWS SDK (p. 1119)
  - Enable an AWS KMS key using an AWS SDK (p. 1121)
  - Encrypt text using an AWS KMS key using an AWS SDK (p. 1124)
  - Generate a plaintext data key for client-side encryption using an AWS SDK (p. 1128)
  - Generate an encrypted data key using an AWS SDK (p. 1129)
  - Get a policy for an AWS KMS key using an AWS SDK (p. 1130)
  - List the aliases defined for an AWS KMS key using an AWS SDK (p. 1130)
  - List grants for an AWS KMS key using an AWS SDK (p. 1133)
  - List AWS KMS keys using an AWS SDK (p. 1135)
  - List policies for an AWS KMS key using an AWS SDK (p. 1138)
  - Reencrypt ciphertext from one AWS KMS key to another using an AWS SDK (p. 1139)
  - Retire a grant for an AWS KMS key using an AWS SDK (p. 1142)
  - Revoke a grant for an AWS KMS key using an AWS SDK (p. 1143)
  - Schedule deletion of an AWS KMS key using an AWS SDK (p. 1143)
  - Set the policy for an AWS KMS key using an AWS SDK (p. 1144)
  - Update the AWS KMS key referred to by an alias using an AWS SDK (p. 1145)
- [Scenarios for AWS KMS using AWS SDKs \(p. 1146\)](#)
  - [Encrypt and decrypt text with AWS KMS keys using an AWS SDK \(p. 1146\)](#)
  - [Manage AWS KMS keys using an AWS SDK \(p. 1148\)](#)

## Actions for AWS KMS using AWS SDKs

The following code examples show how to use AWS Key Management Service with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create a grant for an AWS KMS key using an AWS SDK \(p. 1102\)](#)
- [Create an AWS KMS key using an AWS SDK \(p. 1105\)](#)
- [Generate a random byte string using an AWS SDK \(p. 1109\)](#)
- [Create an alias for an AWS KMS key using an AWS SDK \(p. 1110\)](#)
- [Decrypt ciphertext with an AWS KMS key using an AWS SDK \(p. 1112\)](#)
- [Delete an AWS KMS alias using an AWS SDK \(p. 1116\)](#)
- [Describe an AWS KMS key using an AWS SDK \(p. 1117\)](#)
- [Disable an AWS KMS key using an AWS SDK \(p. 1119\)](#)
- [Enable an AWS KMS key using an AWS SDK \(p. 1121\)](#)
- [Encrypt text using an AWS KMS key using an AWS SDK \(p. 1124\)](#)
- [Generate a plaintext data key for client-side encryption using an AWS SDK \(p. 1128\)](#)
- [Generate an encrypted data key using an AWS SDK \(p. 1129\)](#)
- [Get a policy for an AWS KMS key using an AWS SDK \(p. 1130\)](#)
- [List the aliases defined for an AWS KMS key using an AWS SDK \(p. 1130\)](#)
- [List grants for an AWS KMS key using an AWS SDK \(p. 1133\)](#)
- [List AWS KMS keys using an AWS SDK \(p. 1135\)](#)
- [List policies for an AWS KMS key using an AWS SDK \(p. 1138\)](#)
- [Reencrypt ciphertext from one AWS KMS key to another using an AWS SDK \(p. 1139\)](#)
- [Retire a grant for an AWS KMS key using an AWS SDK \(p. 1142\)](#)
- [Revoke a grant for an AWS KMS key using an AWS SDK \(p. 1143\)](#)
- [Schedule deletion of an AWS KMS key using an AWS SDK \(p. 1143\)](#)
- [Set the policy for an AWS KMS key using an AWS SDK \(p. 1144\)](#)
- [Update the AWS KMS key referred to by an alias using an AWS SDK \(p. 1145\)](#)

## Create a grant for an AWS KMS key using an AWS SDK

The following code examples show how to create a grant for a KMS key.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

This examples grants a user permission to use a key for encryption and decryption.

```
public static async Task Main()
{
    var client = new AmazonKeyManagementServiceClient();

    // The identity that is given permission to perform the operations
```

```
// specified in the grant.
var grantee = "arn:aws:iam::111122223333:role/ExampleRole";

// The identifier of the AWS KMS key to which the grant applies. You
// can use the key ID or the Amazon Resource Name (ARN) of the KMS key.
var keyId = "7c9ecc2-38cb-4c4f-9db3-766ee8dd3ad4";

var request = new CreateGrantRequest
{
    GranteePrincipal = grantee,
    KeyId = keyId,

    // A list of operations that the grant allows.
    Operations = new List<string>
    {
        "Encrypt",
        "Decrypt",
    },
};

var response = await client.CreateGrantAsync(request);

string grantId = response.GrantId; // The unique identifier of the
grant.
string grantToken = response.GrantToken; // The grant token.

Console.WriteLine($"Id: {grantId}, Token: {grantToken}");
}
```

- For API details, see [CreateGrant in AWS SDK for .NET API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createGrant(KmsClient kmsClient, String keyId, String
granteePrincipal, String operation) {

    try {
        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .granteePrincipal(granteePrincipal)
            .operationsWithStrings(operation)
            .build();

        CreateGrantResponse response = kmsClient.createGrant(grantRequest);
        return response.grantId();

    }catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateGrant](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createNewGrant(keyIdVal: String?, granteePrincipalVal: String?,  
operation: String?): String? {  
  
    val operation0b = GrantOperation.fromValue(operation)  
    val grantOperationList = ArrayList<GrantOperation>()  
    grantOperationList.add(operation0b)  
  
    val request = CreateGrantRequest {  
        keyId = keyIdVal  
        granteePrincipal = granteePrincipalVal  
        operations = grantOperationList  
    }  
  
    KmsClient { region = "us-west-2" }.use { kmsClient ->  
        val response = kmsClient.createGrant(request)  
        return response.grantId  
    }  
}
```

- For API details, see [CreateGrant](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

This example grants permission to let a user generate a data key that can be used for encryption.

```
class GrantManager:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
  
    def create_grant(self, key_id):  
        """  
        Creates a grant for a key that lets a user generate a symmetric data  
        encryption key.  
  
        :param key_id: The ARN or ID of the key.  
        :return: The grant that is created.  
        """  
        user = input(  
            f"Enter the ARN of an IAM user to grant that user GenerateDataKey "
```

```
f"permissions on key {key_id}.")
if user != '':
    try:
        grant = self.kms_client.create_grant(
            KeyId=key_id, GranteePrincipal=user,
            Operations=['GenerateDataKey'])
    except ClientError as err:
        logger.error(
            "Couldn't create a grant on key %s. Here's why: %s",
            key_id, err.response['Error']['Message'])
    else:
        print(f"Grant created on key {key_id}.")
        return grant
else:
    print("Skipping grant creation.")
```

- For API details, see [CreateGrant in AWS SDK for Python \(Boto3\) API Reference](#).

## Create an AWS KMS key using an AWS SDK

The following code examples show how to create an AWS KMS key.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

public class CreateKey
{
    public static async Task Main()
    {
        // Note that if you need to create a Key in an AWS Region
        // other than the Region defined for the default user, you need to
        // pass the Region to the client constructor.
        var client = new AmazonKeyManagementServiceClient();

        // The call to CreateKeyAsync will create a symmetrical AWS KMS
        // key. For more information about symmetrical and asymmetrical
        // keys, see:
        //
        // https://docs.aws.amazon.com/kms/latest/developerguide/symm-asymm-
choose.html
        var response = await client.CreateKeyAsync(new CreateKeyRequest());

        // The KeyMetadata object contains information about the new AWS KMS
        // key.
        KeyMetadata keyMetadata = response.KeyMetadata;

        if (keyMetadata is not null)
        {
            Console.WriteLine($"KMS Key: {keyMetadata.KeyId} was successfully
created.");
        }
    }
}
```

```
        else
    {
        Console.WriteLine("Could not create KMS Key.");
    }
}
```

- For API details, see [CreateKey](#) in *AWS SDK for .NET API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/kms"
)

// KMSCreateKeyAPI defines the interface for the CreateKey function.
// We use this interface to test the function using a mocked service.
type KMSCreateKeyAPI interface {
    CreateKey(ctx context.Context,
        params *kms.CreateKeyInput,
        optFns ...func(*kms.Options)) (*kms.CreateKeyOutput, error)
}

// MakeKey creates an AWS Key Management Service (AWS KMS) key (KMS key).
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If success, a CreateKeyOutput object containing the result of the service
//   call and nil.
//   Otherwise, nil and an error from the call to CreateKey.
func MakeKey(c context.Context, api KMSCreateKeyAPI, input *kms.CreateKeyInput)
    (*kms.CreateKeyOutput, error) {
    return api.CreateKey(c, input)
}

func main() {
    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := kms.NewFromConfig(cfg)

    input := &kms.CreateKeyInput{}

    result, err := MakeKey(context.TODO(), client, input)
```

```
if err != nil {
    fmt.Println("Got error creating key:")
    fmt.Println(err)
    return
}

fmt.Println(*result.KeyMetadata.KeyId)
```

- For API details, see [CreateKey](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createKey(KmsClient kmsClient, String keyDesc) {

    try {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
            .keyUsage("ENCRYPT_DECRYPT")
            .build();

        CreateKeyResponse result = kmsClient.createKey(keyRequest);
        System.out.printf("Created a customer key with id \"%s\"\n",
            result.keyMetadata().arn());
        return result.keyMetadata().keyId();
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateKey](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createKey(keyDesc: String?): String? {

    val request = CreateKeyRequest {
```

```
        description = keyDesc
        customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
        keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val result = kmsClient.createKey(request)
        println("Created a customer key with id " + result.keyMetadata?.arn)
        return result.keyMetadata?.keyId
    }
}
```

- For API details, see [CreateKey](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    def create_key(self):
        """
        Creates a key (or multiple keys) with a user-provided description.
        """
        answer = 'y'
        while answer.lower() == 'y':
            key_desc = input("\nLet's create a key. Describe it for me: ")
            if not key_desc:
                key_desc = "Key management demo key"
            try:
                key = self.kms_client.create_key(Description=key_desc)
            except ClientError as err:
                logging.error(
                    "Couldn't create your key. Here's why: %s",
                    err.response['Error']['Message'])
                raise
            else:
                print("Key created:")
                pprint(key)
                self.created_keys.append(key)
            answer = input("Create another (y/n)? ")
```

- For API details, see [CreateKey](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_key(client: &Client) -> Result<(), Error> {
    let resp = client.create_key().send().await?;

    let id = resp
        .key_metadata
        .unwrap()
        .key_id
        .unwrap_or_else(|| String::from("No ID!"));

    println!("Key: {}", id);

    Ok(())
}
```

- For API details, see [CreateKey](#) in *AWS SDK for Rust API reference*.

## Generate a random byte string using an AWS SDK

The following code example shows how to create a random byte string that is cryptographically secure using AWS KMS.

Rust

### SDK for Rust

**Note**

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_string(client: &Client, length: i32) -> Result<(), Error> {
    let resp = client
        .generate_random()
        .number_of_bytes(length)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.plaintext.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}{}", s);

    Ok(())
}
```

- For API details, see [GenerateRandom](#) in *AWS SDK for Rust API reference*.

## Create an alias for an AWS KMS key using an AWS SDK

The following code examples show how to create an alias for a KMS key key.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

public class CreateAlias
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The alias name must start with alias/ and can be
        // up to 256 alphanumeric characters long.
        var aliasName = "alias/ExampleAlias";

        // The value supplied as the TargetKeyId can be either
        // the key ID or key Amazon Resource Name (ARN) of the
        // AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new CreateAliasRequest
        {
            AliasName = aliasName,
            TargetKeyId = keyId,
        };

        var response = await client.CreateAliasAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Alias, {aliasName}, successfully created.");
        }
        else
        {
            Console.WriteLine($"Could not create alias.");
        }
    }
}
```

- For API details, see [CreateAlias in AWS SDK for .NET API Reference](#).

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,  
String aliasName) {  
  
    try {  
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()  
            .aliasName(aliasName)  
            .targetKeyId(targetKeyId)  
            .build();  
  
        kmsClient.createAlias(aliasRequest);  
  
    } catch (KmsException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [CreateAlias in AWS SDK for Java 2.x API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createCustomAlias(targetKeyIdVal: String?, aliasNameVal: String?) {  
  
    val request = CreateAliasRequest {  
        aliasName = aliasNameVal  
        targetKeyId = targetKeyIdVal  
    }  
  
    KmsClient { region = "us-west-2" }.use { kmsClient ->  
        kmsClient.createAlias(request)  
        println("$aliasNameVal was successfully created")  
    }  
}
```

- For API details, see [CreateAlias in AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AliasManager:
```

```
def __init__(self, kms_client):
    self.kms_client = kms_client
    self.created_key = None

def create_alias(self, key_id):
    """
    Creates an alias for the specified key.

    :param key_id: The ARN or ID of a key to give an alias.
    :return: The alias given to the key.
    """
    alias = ''
    while alias == '':
        alias = input(f"What alias would you like to give to key {key_id}? ")
    try:
        self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
    except ClientError as err:
        logger.error(
            "Couldn't create alias %s. Here's why: %s",
            alias, err.response['Error']['Message'])
    else:
        print(f"Created alias {alias} for key {key_id}.")
    return alias
```

- For API details, see [CreateAlias in AWS SDK for Python \(Boto3\) API Reference](#).

## Decrypt ciphertext with an AWS KMS key using an AWS SDK

The following code examples show how to decrypt ciphertext that was encrypted by a KMS key.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    b64 "encoding/base64"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/kms"
)

// KMSDecryptAPI defines the interface for the Decrypt function.
// We use this interface to test the function using a mocked service.
type KMSDecryptAPI interface {
    Decrypt(ctx context.Context,
        params *kms.DecryptInput,
        optFns ...func(*kms.Options)) (*kms.DecryptOutput, error)
}

// DecodeData decrypts some text that was encrypted with an AWS Key Management
// Service (AWS KMS) key (KMS key).
// Inputs:
```

```
//      c is the context of the method call, which includes the AWS Region.
//      api is the interface that defines the method call.
//      input defines the input arguments to the service call.
// Output:
//      If success, a DecryptOutput object containing the result of the service call
//      and nil.
//      Otherwise, nil and an error from the call to Decrypt.
func DecodeData(c context.Context, api KMSDecryptAPI, input *kms.DecryptInput)
(*kms.DecryptOutput, error) {
    return api.Decrypt(c, input)
}

func main() {
    data := flag.String("d", "", "The encrypted data, as a string")
    flag.Parse()

    if *data == "" {
        fmt.Println("You must supply the encrypted data as a string")
        fmt.Println("-d DATA")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := kms.NewFromConfig(cfg)

    blob, err := b64.StdEncoding.DecodeString(*data)
    if err != nil {
        panic("error converting string to blob, " + err.Error())
    }

    input := &kms.DecryptInput{
        CiphertextBlob: blob,
    }

    result, err := DecodeData(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got error decrypting data: ", err)
        return
    }

    fmt.Println(string(result.Plaintext))
}
```

- For API details, see [Decrypt](#) in [AWS SDK for Go API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {

    try {
```

```
DecryptRequest decryptRequest = DecryptRequest.builder()
    .ciphertextBlob(encryptedData)
    .keyId(keyId)
    .build();

DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
decryptResponse.plaintext();

} catch (KmsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [Decrypt](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun encryptData(keyIdValue: String): ByteArray? {

    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest = EncryptRequest {
        keyId = keyIdValue
        plaintext = myBytes
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(encryptedDataVal: ByteArray?, keyIdVal: String?, path: String) {

    val decryptRequest = DecryptRequest {
        ciphertextBlob = encryptedDataVal
        keyId = keyIdVal
    }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Write the decrypted data to a file.
        if (myVal != null) {
            File(path).writeBytes(myVal)
        }
    }
}
```

```
        }  
    }  
}
```

- For API details, see [Decrypt](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyEncrypt:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
  
    def decrypt(self, key_id, cipher_text):  
        """  
        Decrypts text previously encrypted with a key.  
  
        :param key_id: The ARN or ID of the key used to decrypt the data.  
        :param cipher_text: The encrypted text to decrypt.  
        """  
        answer = input("Ready to decrypt your ciphertext (y/n)? ")  
        if answer.lower() == 'y':  
            try:  
                text = self.kms_client.decrypt(  
                    KeyId=key_id, CiphertextBlob=cipher_text)['Plaintext']  
            except ClientError as err:  
                logger.error(  
                    "Couldn't decrypt your ciphertext. Here's why: %s",  
                    err.response['Error']['Message'])  
            else:  
                print(f"Your plaintext is {text.decode()}")  
            else:  
                print("Skipping decryption demo.")
```

- For API details, see [Decrypt](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn decrypt_key(client: &Client, key: &str, filename: &str) -> Result<(),  
Error> {  
    // Open input text file and get contents as a string  
    // input is a base-64 encoded string, so decode it:
```

```
let data = fs::read_to_string(filename)
    .map(|input| {
        base64::decode(input).expect("Input file does not contain valid base 64
characters.")
    })
    .map(Blob::new);

let resp = client
    .decrypt()
    .key_id(key)
    .ciphertext_blob(data.unwrap())
    .send()
    .await?;

let inner = resp.plaintext.unwrap();
let bytes = inner.as_ref();

let s = String::from_utf8(bytes.to_vec()).expect("Could not convert to UTF-8");

println!();
println!("Decoded string:");
println!("{}", s);

Ok(())
}
```

- For API details, see [Decrypt](#) in *AWS SDK for Rust API reference*.

## Delete an AWS KMS alias using an AWS SDK

The following code example shows how to delete an AWS KMS alias.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    def delete_alias(self):
        """
        Deletes an alias.
        """
        alias = input(f"Enter an alias that you'd like to delete: ")
        if alias != '':
            try:
                self.kms_client.delete_alias(AliasName=alias)
            except ClientError as err:
                logger.error(
                    "Couldn't delete alias %s. Here's why: %s",
                    alias, err.response['Error']['Message'])
            else:
                print(f"Deleted alias {alias}.")
        else:
            print("Skipping alias deletion.")
```

- For API details, see [DeleteAlias in AWS SDK for Python \(Boto3\) API Reference](#).

## Describe an AWS KMS key using an AWS SDK

The following code examples show how to describe a KMS key.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

public class DescribeKey
{
    public static async Task Main()
    {
        var keyId = "7c9ecc2-38cb-4c4f-9db3-766ee8dd3ad4";
        var request = new DescribeKeyRequest
        {
            KeyId = keyId,
        };

        var client = new AmazonKeyManagementServiceClient();

        var response = await client.DescribeKeyAsync(request);
        var metadata = response.KeyMetadata;

        Console.WriteLine($"'{metadata.KeyId}' created on:
{metadata.CreationDate}");
        Console.WriteLine($"State: {metadata.KeyState}");
        Console.WriteLine($"'{metadata.Description}'");
    }
}
```

- For API details, see [DescribeKey in AWS SDK for .NET API Reference](#).

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeSpecifcKey(KmsClient kmsClient, String keyId ){

    try {
```

```
DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
    .keyId(keyId)
    .build();

DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
System.out.println("The key description is
"+response.keyMetadata().description());
System.out.println("The key ARN is "+response.keyMetadata().arn());

} catch (KmsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [DescribeKey in AWS SDK for Java 2.x API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeSpecifcKey(keyIdVal: String?) {

    val request = DescribeKeyRequest {
        keyId = keyIdVal
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.describeKey(request)
        println("The key description is ${response.keyMetadata?.description}")
        println("The key ARN is ${response.keyMetadata?.arn}")
    }
}
```

- For API details, see [DescribeKey in AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []
```

```
def describe_key(self):
    """
    Describes a key.
    """
    key_id = input("Enter a key ID or ARN here to get information about the
key: ")
    if key_id:
        try:
            key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']
        except ClientError as err:
            logging.error(
                "Couldn't get key '%s'. Here's why: %s",
                key_id, err.response['Error']['Message'])
    else:
        print(f"Got key {key_id}:")
        pprint(key)
    return key_id
```

- For API details, see [DescribeKey in AWS SDK for Python \(Boto3\) API Reference](#).

## Disable an AWS KMS key using an AWS SDK

The following code examples show how to disable a KMS key.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

public class DisableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to disable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new DisableKeyRequest
        {
            KeyId = keyId,
        };

        var response = await client.DisableKeyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            // Retrieve information about the key to show that it has now
            // been disabled.
            var describeResponse = await client.DescribeKeyAsync(new
DescribeKeyRequest
{
```

```
       KeyId = keyId,
    });
    Console.WriteLine($"'{describeResponse.KeyMetadata.KeyId}' - state:
{describeResponse.KeyMetadata.KeyState}");
}
}
```

- For API details, see [DisableKey](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void disableKey( KmsClient kmsClient, String keyId) {

    try {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.disableKey(keyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DisableKey](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun disableKey(keyIdVal: String?) {

    val request = DisableKeyRequest {
        keyId = keyIdVal
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        kmsClient.disableKey(request)
        println("$keyIdVal was successfully disabled")
    }
}
```

```
    }  
}
```

- For API details, see [DisableKey](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyManager:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
        self.created_keys = []  
  
    def enable_disable_key(self, key_id):  
        """  
        Disables and then enables a key. Gets the key state after each state  
        change.  
        """  
        answer = input("Do you want to disable and then enable that key (y/n)? ")  
        if answer.lower() == 'y':  
            try:  
                self.kms_client.disable_key(KeyId=key_id)  
                key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']  
            except ClientError as err:  
                logging.error(  
                    "Couldn't disable key '%s'. Here's why: %s",  
                    key_id, err.response['Error']['Message'])  
            else:  
                print(f"AWS KMS says your key state is: {key['KeyState']}")  
  
            try:  
                self.kms_client.enable_key(KeyId=key_id)  
                key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']  
            except ClientError as err:  
                logging.error(  
                    "Couldn't enable key '%s'. Here's why: %s",  
                    key_id, err.response['Error']['Message'])  
            else:  
                print(f"AWS KMS says your key state is: {key['KeyState']}")
```

- For API details, see [DisableKey](#) in *AWS SDK for Python (Boto3) API Reference*.

## Enable an AWS KMS key using an AWS SDK

The following code examples show how to enable a KMS key.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

public class EnableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to enable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new EnableKeyRequest
        {
            KeyId = keyId,
        };

        var response = await client.EnableKeyAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            // Retrieve information about the key to show that it has now
            // been enabled.
            var describeResponse = await client.DescribeKeyAsync(new
DescribeKeyRequest
{
    KeyId = keyId,
});
Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
{describeResponse.KeyMetadata.KeyState}");
        }
    }
}
```

- For API details, see [EnableKey](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void enableKey(KmsClient kmsClient, String keyId) {

    try {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKey(enableKeyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}
```

- For API details, see [EnableKey](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun enableKey(keyIdVal: String?) {  
  
    val request = EnableKeyRequest {  
        keyId = keyIdVal  
    }  
  
    KmsClient { region = "us-west-2" }.use { kmsClient ->  
        kmsClient.enableKey(request)  
        println("$keyIdVal was successfully enabled.")  
    }  
}
```

- For API details, see [EnableKey](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyManager:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
        self.created_keys = []  
  
    def enable_disable_key(self, key_id):  
        """  
        Disables and then enables a key. Gets the key state after each state  
        change.  
        """  
        answer = input("Do you want to disable and then enable that key (y/n)? ")  
        if answer.lower() == 'y':  
            try:  
                self.kms_client.disable_key(KeyId=key_id)  
                key = self.kms_client.describe_key(KeyId=key_id)[('KeyMetadata')]  
            except ClientError as err:  
                logging.error(  
                    "Couldn't disable key '%s'. Here's why: %s",  
                    key_id, err.response['Error']['Message'])
```

```
        else:
            print(f"AWS KMS says your key state is: {key['KeyState']}.")

        try:
            self.kms_client.enable_key(KeyId=key_id)
            key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']
        except ClientError as err:
            logging.error(
                "Couldn't enable key '%s'. Here's why: %s",
                key_id, err.response['Error']['Message'])
        else:
            print(f"AWS KMS says your key state is: {key['KeyState']}")
```

- For API details, see [EnableKey](#) in *AWS SDK for Python (Boto3) API Reference*.

## Encrypt text using an AWS KMS key using an AWS SDK

The following code examples show how to encrypt text using a KMS key.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    b64 "encoding/base64"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/kms"
)

// KMSEncryptAPI defines the interface for the Encrypt function.
// We use this interface to test the function using a mocked service.
type KMSEncryptAPI interface {
    Encrypt(ctx context.Context,
        params *kms.EncryptInput,
        optFns ...func(*kms.Options)) (*kms.EncryptOutput, error)
}

// EncryptText encrypts some text using an AWS Key Management Service (AWS KMS) key
// (KMS key).
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If success, an EncryptOutput object containing the result of the service
//     call and nil.
//     Otherwise, nil and an error from the call to Encrypt.
func EncryptText(c context.Context, api KMSEncryptAPI, input *kms.EncryptInput)
    (*kms.EncryptOutput, error) {
    return api.Encrypt(c, input)
}
```

```
func main() {
    keyID := flag.String("k", "", "The ID of a KMS key")
    text := flag.String("t", "", "The text to encrypt")
    flag.Parse()

    if *keyID == "" || *text == "" {
        fmt.Println("You must supply the ID of a KMS key and some text")
        fmt.Println("-k KEY-ID -t \"text\"")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := kms.NewFromConfig(cfg)

    input := &kms.EncryptInput{
        KeyId:     keyID,
        Plaintext: []byte(*text),
    }

    result, err := EncryptText(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got error encrypting data:")
        fmt.Println(err)
        return
    }

    blobString := b64.StdEncoding.EncodeToString(result.CiphertextBlob)

    fmt.Println(blobString)
}
```

- For API details, see [Encrypt](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static SdkBytes encryptData(KmsClient kmsClient, String keyId) {

    try {
        SdkBytes myBytes = SdkBytes.fromByteArray(new byte[]{1, 2, 3, 4, 5, 6,
7, 8, 9, 0});

        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        EncryptResponse response = kmsClient.encrypt(encryptRequest);
        String algorithm = response.encryptionAlgorithm().toString();
        System.out.println("The encryption algorithm is " + algorithm);
    }
}
```

```
// Get the encrypted data.  
SdkBytes encryptedData = response.ciphertextBlob();  
return encryptedData;  
  
} catch (KmsException e) {  
    System.err.println(e.getMessage());  
    System.exit(1);  
}  
return null;  
}
```

- For API details, see [Encrypt](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun encryptData(keyIdValue: String): ByteArray? {  
  
    val text = "This is the text to encrypt by using the AWS KMS Service"  
    val myBytes: ByteArray = text.toByteArray()  
  
    val encryptRequest = EncryptRequest {  
        keyId = keyIdValue  
        plaintext = myBytes  
    }  
  
    KmsClient { region = "us-west-2" }.use { kmsClient ->  
        val response = kmsClient.encrypt(encryptRequest)  
        val algorithm: String = response.encryptionAlgorithm.toString()  
        println("The encryption algorithm is $algorithm")  
  
        // Return the encrypted data.  
        return response.ciphertextBlob  
    }  
}  
  
suspend fun decryptData(encryptedDataVal: ByteArray?, keyIdVal: String?, path: String) {  
  
    val decryptRequest = DecryptRequest {  
        ciphertextBlob = encryptedDataVal  
        keyId = keyIdVal  
    }  
    KmsClient { region = "us-west-2" }.use { kmsClient ->  
        val decryptResponse = kmsClient.decrypt(decryptRequest)  
        val myVal = decryptResponse.plaintext  
  
        // Write the decrypted data to a file.  
        if (myVal != null) {  
            File(path).writeBytes(myVal)  
        }  
    }  
}
```

- For API details, see [Encrypt](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyEncrypt:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
  
    def encrypt(self, key_id):  
        """  
        Encrypts text by using the specified key.  
  
        :param key_id: The ARN or ID of the key to use for encryption.  
        :return: The encrypted version of the text.  
        """  
        text = input("Enter some text to encrypt: ")  
        try:  
            cipher_text = self.kms_client.encrypt(  
                KeyId=key_id, Plaintext=text.encode())['CiphertextBlob']  
        except ClientError as err:  
            logger.error(  
                "Couldn't encrypt text. Here's why: %s", err.response['Error']  
                ['Message'])  
        else:  
            print(f"Your ciphertext is: {cipher_text}")  
            return cipher_text
```

- For API details, see [Encrypt](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn encrypt_string(  
    verbose: bool,  
    client: &Client,  
    text: &str,  
    key: &str,  
    out_file: &str,  
) -> Result<(), Error> {  
    let blob = Blob::new(text.as_bytes());  
  
    let resp = client.encrypt().key_id(key).plaintext(blob).send().await?;
```

```
// Did we get an encrypted blob?  
let blob = resp.ciphertext_blob.expect("Could not get encrypted text");  
let bytes = blob.as_ref();  
  
let s = base64::encode(bytes);  
  
let mut ofile = File::create(out_file).expect("unable to create file");  
ofile.write_all(s.as_bytes()).expect("unable to write");  
  
if verbose {  
    println!("Wrote the following to {:?}", out_file);  
    println!("{} ", s);  
}  
  
Ok(())  
}
```

- For API details, see [Encrypt](#) in *AWS SDK for Rust API reference*.

## Generate a plaintext data key for client-side encryption using an AWS SDK

The following code examples show how to generate a unique symmetric data key for client-side encryption from an AWS KMS key. The key contains both plaintext and ciphertext versions.

### Python

#### [SDK for Python \(Boto3\)](#)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyManager:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
        self.created_keys = []  
  
    def generate_data_key(self, key_id):  
        """  
        Generates a symmetric data key that can be used for client-side encryption.  
        """  
        answer = input(  
            f"Do you want to generate a symmetric data key from key {key_id} (y/n)?")  
        if answer.lower() == 'y':  
            try:  
                data_key = self.kms_client.generate_data_key(KeyId=key_id,  
KeySpec='AES_256')  
            except ClientError as err:  
                logger.error(  
                    "Couldn't generate a data key for key %s. Here's why: %s",  
                    key_id, err.response['Error']['Message'])  
            else:  
                pprint(data_key)
```

- For API details, see [GenerateDataKey](#) in *AWS SDK for Python (Boto3) API Reference*.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}{}", s);

    Ok(())
}
```

- For API details, see [GenerateDataKey](#) in *AWS SDK for Rust API reference*.

## Generate an encrypted data key using an AWS SDK

The following code example shows how to generate an encrypted data key from an AWS KMS key. The key contains only a ciphertext version.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key_without_plaintext()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;
```

```
// Did we get an encrypted blob?  
let blob = resp.ciphertext_blob.expect("Could not get encrypted text");  
let bytes = blob.as_ref();  
  
let s = base64::encode(bytes);  
  
println!();  
println!("Data key:");  
println!("{}", s);  
  
Ok(())
}
```

- For API details, see [GenerateDataKeyWithoutPlaintext](#) in *AWS SDK for Rust API reference*.

## Get a policy for an AWS KMS key using an AWS SDK

The following code example shows how to get a policy by name for a KMS key.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyPolicy:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
  
    def get_policy(self, key_id):  
        """  
        Gets the policy of a key.  
  
        :param key_id: The ARN or ID of the key to query.  
        :return: The key policy as a dict.  
        """  
        if key_id != '':  
            try:  
                response = self.kms_client.get_key_policy(  
                    KeyId=key_id, PolicyName='default')  
                policy = json.loads(response['Policy'])  
            except ClientError as err:  
                logger.error(  
                    "Couldn't get policy for key %s. Here's why: %s",  
                    key_id, err.response['Error']['Message'])  
            else:  
                pprint(policy)  
                return policy  
        else:  
            print("Skipping get policy demo.")
```

- For API details, see [GetKeyPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## List the aliases defined for an AWS KMS key using an AWS SDK

The following code examples show how to list aliases for a KMS key.

.NET

**AWS SDK for .NET**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

public class ListAliases
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();
        var request = new ListAliasesRequest();
        var response = new ListAliasesResponse();

        do
        {
            response = await client.ListAliasesAsync(request);

            response.Aliases.ForEach(alias =>
            {
                Console.WriteLine($"Created: {alias.CreationDate} Last Update: {alias.LastUpdatedDate} Name: {alias.AliasName}");
            });

            request.Marker = response.NextMarker;
        }
        while (response.Truncated);
    }
}
```

- For API details, see [ListAliases](#) in *AWS SDK for .NET API Reference*.

Java

**SDK for Java 2.x**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllAliases( KmsClient kmsClient ) {

    try {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesResponse aliasesResponse =
        kmsClient.listAliases(aliasesRequest) ;
        List<AliasListEntry> aliases = aliasesResponse_aliases();
        for (AliasListEntry alias: aliases) {
            System.out.println("The alias name is: "+alias.aliasName());
    }
}
```

```
        }

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListAliases](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAllAliases() {

    val request = ListAliasesRequest {
        limit = 15
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
        response.aliases?.forEach { alias ->
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

- For API details, see [ListAliases](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    def list_aliases(self):
        """
        Lists aliases for the current account.
        """
        answer = input("\nLet's list your key aliases. Ready (y/n)? ")
        if answer.lower() == 'y':
```

```
try:
    page_size = 10

    alias Paginator = self.kms_client.getPaginator('list_aliases')
    for alias_page in
aliasPaginator.paginate(PaginationConfig={'PageSize': 10}):
        print(f"Here are {page_size} aliases:")
        pprint(alias_page['Aliases'])
        if alias_page['Truncated']:
            answer = input(
                f"Do you want to see the next {page_size} aliases (y/n)? ")
            if answer.lower() != 'y':
                break
            else:
                print("That's all your aliases!")
    except ClientError as err:
        logging.error(
            "Couldn't list your aliases. Here's why: %s",
            err.response['Error']['Message'])
```

- For API details, see [ListAliases](#) in *AWS SDK for Python (Boto3) API Reference*.

## List grants for an AWS KMS key using an AWS SDK

The following code examples show how to list grants for a KMS key.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

public class ListGrants
{
    public static async Task Main()
    {
        // The identifier of the AWS KMS key to disable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";
        var client = new AmazonKeyManagementServiceClient();
        var request = new ListGrantsRequest
        {
            KeyId = keyId,
        };

        var response = new ListGrantsResponse();

        do
        {
            response = await client.ListGrantsAsync(request);

            response.Grants.ForEach(grant =>
            {

```

```
        Console.WriteLine($"{{grant.GrantId}}");
    });

    request.Marker = response.NextMarker;
}
while (response.Truncated);
}
```

- For API details, see [ListGrants](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void displayGrantIds(KmsClient kmsClient, String keyId) {

    try {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsResponse response = kmsClient.listGrants(grantsRequest);
        List<GrantListEntry> grants = response.grants();
        for ( GrantListEntry grant: grants) {
            System.out.println("The grant Id is : "+grant.grantId());
        }
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListGrants](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun displayGrantIds(keyIdVal: String?) {

    val request = ListGrantsRequest {
        keyId = keyIdVal
    }
}
```

```
        limit = 15
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- For API details, see [ListGrants](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    def list_grants(self, key_id):
        """
        Lists grants for a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The grants for the key.
        """
        answer = input(f"Ready to list grants on key {key_id} (y/n)? ")
        if answer.lower() == 'y':
            try:
                grants = self.kms_client.list_grants(KeyId=key_id)['Grants']
            except ClientError as err:
                logger.error(
                    "Couldn't list grants for key %s. Here's why: %s",
                    key_id, err.response['Error']['Message'])
            else:
                print(f"Grants for key {key_id}:")
                pprint(grants)
                return grants
        
```

- For API details, see [ListGrants](#) in *AWS SDK for Python (Boto3) API Reference*.

## List AWS KMS keys using an AWS SDK

The following code examples show how to list KMS keys.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

public class ListKeys
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();
        var request = new ListKeysRequest();
        var response = new ListKeysResponse();

        do
        {
            response = await client.ListKeysAsync(request);

            response.Keys.ForEach(key =>
            {
                Console.WriteLine($"ID: {key.KeyId}, {key.KeyArn}");
            });

            // Set the Marker property when response.Truncated is true
            // in order to get the next keys.
            request.Marker = response.NextMarker;
        }
        while (response.Truncated);
    }
}
```

- For API details, see [ListKeys](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllKeys(KmsClient kmsClient) {

    try {
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        ListKeysResponse keysResponse = kmsClient.listKeys(listKeysRequest);
        List<KeyListEntry> keyListEntries = keysResponse.keys();
        for (KeyListEntry key : keyListEntries) {
            System.out.println("The key ARN is: " + key.keyArn());
            System.out.println("The key Id is: " + key.keyId());
        }
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListKeys](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAllKeys() {  
  
    val request = ListKeysRequest {  
        limit = 15  
    }  
  
    KmsClient { region = "us-west-2" }.use { kmsClient ->  
        val response = kmsClient.listKeys(request)  
        response.keys?.forEach { key ->  
            println("The key ARN is ${key.keyArn}")  
            println("The key Id is ${key.keyId}")  
        }  
    }  
}
```

- For API details, see [ListKeys](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyManager:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
        self.created_keys = []  
  
    def list_keys(self):  
        """  
        Lists the keys for the current account by using a paginator.  
        """  
        try:  
            page_size = 10  
            print("\nLet's list your keys.")  
            keyPaginator = self.kms_client.get_paginator('list_keys')  
            for key_page in keyPaginator.paginate(PaginationConfig={'PageSize':  
10}):  
                print(f"Here are {len(key_page['Keys'])} keys:")
```

```
        pprint(key_page['Keys'])
        if key_page['Truncated']:
            answer = input(f"Do you want to see the next {page_size} keys
(y/n)? ")
            if answer.lower() != 'y':
                break
            else:
                print("That's all your keys!")
        except ClientError as err:
            logging.error(
                "Couldn't list your keys. Here's why: %s", err.response['Error']
                ['Message'])
```

- For API details, see [ListKeys](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_keys(client: &Client) -> Result<(), Error> {
    let resp = client.list_keys().send().await?;

    let keys = resp.keys.unwrap_or_default();

    let len = keys.len();

    for key in keys {
        println!("Key ARN: {}", key.key_arn.as_deref().unwrap_or_default());
    }

    println!();
    println!("Found {} keys", len);

    Ok(())
}
```

- For API details, see [ListKeys](#) in *AWS SDK for Rust API reference*.

## List policies for an AWS KMS key using an AWS SDK

The following code example shows how to list policies for a KMS key.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    def list_policies(self, key_id):
        """
        Lists the names of the policies for a key.

        :param key_id: The ARN or ID of the key to query.
        """
        try:
            policy_names = self.kms_client.list_key_policies(KeyId=key_id)[
                'PolicyNames'
            ]
        except ClientError as err:
            logging.error(
                "Couldn't list your policies. Here's why: %s",
                err.response['Error']['Message']
            )
        else:
            print(f"The policies for key {key_id} are:")
            pprint(policy_names)
```

- For API details, see [ListKeyPolicies](#) in *AWS SDK for Python (Boto3) API Reference*.

## Reencrypt ciphertext from one AWS KMS key to another using an AWS SDK

The following code examples show how to reencrypt ciphertext from one KMS key to another.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/kms"
)

// KMSReEncryptAPI defines the interface for the ReEncrypt function.
// We use this interface to test the function using a mocked service.
type KMSReEncryptAPI interface {
    ReEncrypt(ctx context.Context,
        params *kms.ReEncryptInput,
        optFns ...func(*kms.Options)) (*kms.ReEncryptOutput, error)
}

// ReEncryptText reencrypts some text using a new AWS Key Management Service (AWS
// KMS) key (KMS key).
// Inputs:
```

```
//      c is the context of the method call, which includes the AWS Region.
//      api is the interface that defines the method call.
//      input defines the input arguments to the service call.
// Output:
//      If success, a ReEncryptOutput object containing the result of the service
//      call and nil.
//      Otherwise, nil and an error from the call to ReEncrypt.
func ReEncryptText(c context.Context, api KMSReEncryptAPI, input
    *kms.ReEncryptInput) (*kms.ReEncryptOutput, error) {
    return api.ReEncrypt(c, input)
}

func main() {
    keyID := flag.String("k", "", "The ID of a KMS key")
    data := flag.String("d", "", "The data to reencrypt, as a string")
    flag.Parse()

    if *keyID == "" || *data == "" {
        fmt.Println("You must supply the ID of a KMS key and data")
        fmt.Println("-k KEY-ID -d DATA")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := kms.NewFromConfig(cfg)

    blob := []byte(*data)

    input := &kms.ReEncryptInput{
        CiphertextBlob: blob,
        DestinationKeyId: keyID,
    }

    result, err := ReEncryptText(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got error reencrypting data:")
        fmt.Println(err)
        return
    }

    fmt.Println("Blob (base-64 byte array):")
    fmt.Println(result.CiphertextBlob)
}
```

- For API details, see [ReEncrypt](#) in *AWS SDK for Go API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyEncrypt:
    def __init__(self, kms_client):
```

```

        self.kms_client = kms_client

def re_encrypt(self, source_key_id, cipher_text):
    """
    Takes ciphertext previously encrypted with one key and reencrypt it by
    using another key.

    :param source_key_id: The ARN or ID of the original key used to encrypt the
                         ciphertext.
    :param cipher_text: The encrypted ciphertext.
    :return: The ciphertext encrypted by the second key.
    """
    destination_key_id = input(
        f"Your ciphertext is currently encrypted with key {source_key_id}. "
        f"Enter another key ID or ARN to reencrypt it: ")
    if destination_key_id != '':
        try:
            cipher_text = self.kms_client.re_encrypt(
                SourceKeyId=source_key_id, DestinationKeyId=destination_key_id,
                CiphertextBlob=cipher_text)['CiphertextBlob']
        except ClientError as err:
            logger.error(
                "Couldn't reencrypt your ciphertext. Here's why: %s",
                err.response['Error']['Message'])
        else:
            print(f"Reencrypted your ciphertext as: {cipher_text}")
            return cipher_text
    else:
        print("Skipping reencryption demo.")

```

- For API details, see [ReEncrypt](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

async fn reencrypt_string(
    verbose: bool,
    client: &Client,
    input_file: &str,
    output_file: &str,
    first_key: &str,
    new_key: &str,
) -> Result<(), Error> {
    // Get blob from input file
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(input_file)
        .map(|input_file| base64::decode(input_file).expect("invalid base 64"))
        .map(Blob::new);

    let resp = client

```

```
.re_encrypt()
.ciphertext_blob(data.unwrap())
.source_key_id(first_key)
.destination_key_id(new_key)
.send()
.await?;

// Did we get an encrypted blob?
let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
let bytes = blob.as_ref();

let s = base64::encode(bytes);
let o = &output_file;

let mut ofile = File::create(o).expect("unable to create file");
ofile.write_all(s.as_bytes()).expect("unable to write");

if verbose {
    println!("Wrote the following to {}:", output_file);
    println!("{}", s);
} else {
    println!("Wrote base64-encoded output to {}", output_file);
}

Ok(())
}
```

- For API details, see [ReEncrypt](#) in *AWS SDK for Rust API reference*.

## Retire a grant for an AWS KMS key using an AWS SDK

The following code example shows how to retire a grant for a KMS key.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    def retire_grant(self, grant):
        """
        Retires a grant so that it can no longer be used.

        :param grant: The grant to retire.
        """
        try:
            self.kms_client.retire_grant(GrantToken=grant['GrantToken'])
        except ClientError as err:
            logger.error(
                "Couldn't retire grant %s. Here's why: %s",
                grant['GrantId'], err.response['Error']['Message'])
        else:
            print(f"Grant {grant['GrantId']} retired.")
```

- For API details, see [RetireGrant in AWS SDK for Python \(Boto3\) API Reference](#).

## Revoke a grant for an AWS KMS key using an AWS SDK

The following code example shows how to revoke a grant for a KMS key.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GrantManager:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
  
    def revoke_grant(self, key_id, grant):  
        """  
        Revokes a grant so that it can no longer be used.  
  
        :param key_id: The ARN or ID of the key associated with the grant.  
        :param grant: The grant to revoke.  
        """  
        try:  
            self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant['GrantId'])  
        except ClientError as err:  
            logger.error(  
                "Couldn't revoke grant %s. Here's why: %s",  
                grant['GrantId'], err.response['Error']['Message'])  
        else:  
            print(f"Grant {grant['GrantId']} revoked.")
```

- For API details, see [RevokeGrant in AWS SDK for Python \(Boto3\) API Reference](#).

## Schedule deletion of an AWS KMS key using an AWS SDK

The following code example shows how to schedule deletion of a KMS key.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyManager:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
        self.created_keys = []  
  
    def delete_keys(self, keys):  
        """  
        Deletes a list of keys.
```

```
:param keys: The list of keys to delete.  
"""  
answer = input("Do you want to delete these keys (y/n)? ")  
if answer.lower() == 'y':  
    window = 7  
    for key in keys:  
        try:  
            self.kms_client.schedule_key_deletion(  
                KeyId=key['KeyId'], PendingWindowInDays=window)  
        except ClientError as err:  
            logging.error(  
                "Couldn't delete key %s. Here's why: %s",  
                key['KeyId'], err.response['Error']['Message'])  
        else:  
            print(f"Key {key['KeyId']} scheduled for deletion in {window}  
days.")
```

- For API details, see [ScheduleKeyDeletion in AWS SDK for Python \(Boto3\) API Reference](#).

## Set the policy for an AWS KMS key using an AWS SDK

The following code example shows how to set the policy for a KMS key.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyPolicy:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
  
    def set_policy(self, key_id, policy):  
        """  
        Sets the policy of a key. Setting a policy entirely overwrites the existing  
        policy, so care is taken to add a statement to the existing list of  
        statements  
        rather than simply writing a new policy.  
  
        :param key_id: The ARN or ID of the key to set the policy to.  
        :param policy: The existing policy of the key.  
        """  
        user = input("Enter the ARN of an IAM user to set as the principal on the  
policy: ")  
        if key_id != '' and user != '':  
            # The updated policy replaces the existing policy. Add a new statement  
            to  
            # the list along with the original policy statements.  
            policy['Statement'].append({  
                "Sid": "Allow access for ExampleUser",  
                "Effect": "Allow",  
                "Principal": {"AWS": user},  
                "Action": [  
                    "kms:Encrypt",  
                    "kms:GenerateDataKey*",  
                    "kms:Decrypt",
```

```
        "kms:DescribeKey",
        "kms:ReEncrypt*"],
    "Resource": "*"})
try:
    self.kms_client.put_key_policy(
        KeyId=key_id, PolicyName='default', Policy=json.dumps(policy))
except ClientError as err:
    logger.error(
        "Couldn't set policy for key %s. Here's why %s",
        key_id, err.response['Error']['Message'])
else:
    print(f"Set policy for key {key_id}.")
else:
    print("Skipping set policy demo.")
```

- For API details, see [PutKeyPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update the AWS KMS key referred to by an alias using an AWS SDK

The following code example shows how to update the KMS key referred to by an alias.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    def update_alias(self, alias, current_key_id):
        """
        Updates an alias by assigning it to another key.

        :param alias: The alias to reassign.
        :param current_key_id: The ARN or ID of the key currently associated with
        the alias.
        """
        new_key_id = input(
            f"Alias {alias} is currently associated with {current_key_id}. "
            f"Enter another key ID or ARN that you want to associate with {alias}:")
        if new_key_id != '':
            try:
                self.kms_client.update_alias(AliasName=alias,
                    TargetKeyId=new_key_id)
            except ClientError as err:
                logger.error(
                    "Couldn't associate alias %s with key %s. Here's why: %s",
                    alias, new_key_id, err.response['Error']['Message'])
            else:
                print(f"Alias {alias} is now associated with key {new_key_id}.")
        else:
            print("Skipping alias update.")
```

- For API details, see [UpdateAlias in AWS SDK for Python \(Boto3\) API Reference](#).

## Scenarios for AWS KMS using AWS SDKs

The following code examples show how to use AWS Key Management Service with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Encrypt and decrypt text with AWS KMS keys using an AWS SDK \(p. 1146\)](#)
- [Manage AWS KMS keys using an AWS SDK \(p. 1148\)](#)

## Encrypt and decrypt text with AWS KMS keys using an AWS SDK

The following code example shows how to:

- Encrypt plain text by using a KMS key.
- Decrypt ciphertext by using a KMS key.
- Reencrypt ciphertext by using a second KMS key.

### Python

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    def encrypt(self, key_id):
        """
        Encrypts text by using the specified key.

        :param key_id: The ARN or ID of the key to use for encryption.
        :return: The encrypted version of the text.
        """
        text = input("Enter some text to encrypt: ")
        try:
            cipher_text = self.kms_client.encrypt(
               KeyId=key_id, Plaintext=text.encode())['CiphertextBlob']
        except ClientError as err:
            logger.error(
                "Couldn't encrypt text. Here's why: %s", err.response['Error']
            ['Message'])
```

```

        else:
            print(f"Your ciphertext is: {cipher_text}")
            return cipher_text

    def decrypt(self, key_id, cipher_text):
        """
        Decrypts text previously encrypted with a key.

        :param key_id: The ARN or ID of the key used to decrypt the data.
        :param cipher_text: The encrypted text to decrypt.
        """
        answer = input("Ready to decrypt your ciphertext (y/n)? ")
        if answer.lower() == 'y':
            try:
                text = self.kms_client.decrypt(
                    KeyId=key_id, CiphertextBlob=cipher_text)['Plaintext']
            except ClientError as err:
                logger.error(
                    "Couldn't decrypt your ciphertext. Here's why: %s",
                    err.response['Error']['Message'])
            else:
                print(f"Your plaintext is {text.decode()}")
        else:
            print("Skipping decryption demo.")

    def re_encrypt(self, source_key_id, cipher_text):
        """
        Takes ciphertext previously encrypted with one key and reencrypt it by
        using another key.

        :param source_key_id: The ARN or ID of the original key used to encrypt the
                             ciphertext.
        :param cipher_text: The encrypted ciphertext.
        :return: The ciphertext encrypted by the second key.
        """
        destination_key_id = input(
            f"Your ciphertext is currently encrypted with key {source_key_id}. "
            f"Enter another key ID or ARN to reencrypt it: ")
        if destination_key_id != '':
            try:
                cipher_text = self.kms_client.re_encrypt(
                    SourceKeyId=source_key_id, DestinationKeyId=destination_key_id,
                    CiphertextBlob=cipher_text)['CiphertextBlob']
            except ClientError as err:
                logger.error(
                    "Couldn't reencrypt your ciphertext. Here's why: %s",
                    err.response['Error']['Message'])
            else:
                print(f"Reencrypted your ciphertext as: {cipher_text}")
                return cipher_text
        else:
            print("Skipping reencryption demo.")

    def key_encryption(kms_client):
        logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

        print('*'*88)
        print("Welcome to the AWS Key Management Service (AWS KMS) key encryption"
              "demo.")
        print('*'*88)

        key_id = input("Enter a key ID or ARN to start the demo: ")
        if key_id == '':
            print("A key is required to run this demo.")

```

```
    return

key_encrypt = KeyEncrypt(kms_client)
cipher_text = key_encrypt.encrypt(key_id)
print('*'*88)
if cipher_text is not None:
    key_encrypt.decrypt(key_id, cipher_text)
    print('*'*88)
    key_encrypt.re_encrypt(key_id, cipher_text)

print("\nThanks for watching!")
print('*'*88)

if __name__ == '__main__':
    try:
        key_encryption(boto3.client('kms'))
    except Exception:
        logging.exception("Something went wrong with the demo!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [Decrypt](#)
  - [Encrypt](#)
  - [ReEncrypt](#)

## Manage AWS KMS keys using an AWS SDK

The following code example shows how to:

- Create a KMS key.
- List KMS keys for your account and get details about them.
- Enable and disable KMS keys.
- Generate a symmetric data key that can be used for client-side encryption.
- Delete KMS keys.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []
```

```
def create_key(self):
    """
    Creates a key (or multiple keys) with a user-provided description.
    """
    answer = 'y'
    while answer.lower() == 'y':
        key_desc = input("\nLet's create a key. Describe it for me: ")
        if not key_desc:
            key_desc = "Key management demo key"
        try:
            key = self.kms_client.create_key(Description=key_desc)
        except ClientError as err:
            logging.error(
                "Couldn't create your key. Here's why: %s",
                err.response['Error']['Message'])
            raise
        else:
            print("Key created:")
            pprint(key)
            self.created_keys.append(key)
            answer = input("Create another (y/n)? ")

def list_keys(self):
    """
    Lists the keys for the current account by using a paginator.
    """
    try:
        page_size = 10
        print("\nLet's list your keys.")
        keyPaginator = self.kms_client.get_paginator('list_keys')
        for key_page in keyPaginator.paginate(PaginationConfig={'PageSize': 10}):
            print(f"Here are {len(key_page['Keys'])} keys:")
            pprint(key_page['Keys'])
            if key_page['Truncated']:
                answer = input(f"Do you want to see the next {page_size} keys (y/n)? ")
                if answer.lower() != 'y':
                    break
            else:
                print("That's all your keys!")
    except ClientError as err:
        logging.error(
            "Couldn't list your keys. Here's why: %s", err.response['Error']['Message'])

    def describe_key(self):
        """
        Describes a key.
        """
        key_id = input("Enter a key ID or ARN here to get information about the key: ")
        if key_id:
            try:
                key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']
            except ClientError as err:
                logging.error(
                    "Couldn't get key '%s'. Here's why: %s",
                    key_id, err.response['Error']['Message'])
            else:
                print(f"Got key {key_id}:")
                pprint(key)
        return key_id

    def generate_data_key(self, key_id):
```

```

"""
Generates a symmetric data key that can be used for client-side encryption.
"""
answer = input(
    f"Do you want to generate a symmetric data key from key {key_id} (y/n)?"
)
if answer.lower() == 'y':
    try:
        data_key = self.kms_client.generate_data_key(KeyId=key_id,
KeySpec='AES_256')
    except ClientError as err:
        logger.error(
            "Couldn't generate a data key for key %s. Here's why: %s",
            key_id, err.response['Error']['Message'])
    else:
        pprint(data_key)

def enable_disable_key(self, key_id):
"""
Disables and then enables a key. Gets the key state after each state
change.
"""
answer = input("Do you want to disable and then enable that key (y/n)? ")
if answer.lower() == 'y':
    try:
        self.kms_client.disable_key(KeyId=key_id)
        key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']
    except ClientError as err:
        logging.error(
            "Couldn't disable key '%s'. Here's why: %s",
            key_id, err.response['Error']['Message'])
    else:
        print(f"AWS KMS says your key state is: {key['KeyState']}.")

    try:
        self.kms_client.enable_key(KeyId=key_id)
        key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']
    except ClientError as err:
        logging.error(
            "Couldn't enable key '%s'. Here's why: %s",
            key_id, err.response['Error']['Message'])
    else:
        print(f"AWS KMS says your key state is: {key['KeyState']}.")

def delete_keys(self, keys):
"""
Deletes a list of keys.

:param keys: The list of keys to delete.
"""
answer = input("Do you want to delete these keys (y/n)? ")
if answer.lower() == 'y':
    window = 7
    for key in keys:
        try:
            self.kms_client.schedule_key_deletion(
                KeyId=key['KeyId'], PendingWindowInDays=window)
        except ClientError as err:
            logging.error(
                "Couldn't delete key %s. Here's why: %s",
                key['KeyId'], err.response['Error']['Message'])
        else:
            print(f"Key {key['KeyId']} scheduled for deletion in {window}
days.")

```

```
def key_management(kms_client):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the AWS Key Management Service (AWS KMS) key management
demo.")
    print('*'*88)

    key_manager = KeyManager(kms_client)
    key_manager.create_key()
    print('*'*88)
    key_manager.list_keys()
    print('*'*88)
    key_id = key_manager.describe_key()
    if key_id:
        key_manager.enable_disable_key(key_id)
        print('*'*88)
        key_manager.generate_data_key(key_id)
    print('*'*88)
    print("For this demo, we created these keys:")
    for key in key_manager.created_keys:
        print(f"\tKeyId: {key['KeyId']}")
        print(f"\tDescription: {key['Description']}")
        print('*'*66)
    key_manager.delete_keys(key_manager.created_keys)
    print("\nThanks for watching!")
    print('*'*88)

if __name__ == '__main__':
    try:
        key_management(boto3.client('kms'))
    except Exception:
        logging.exception("Something went wrong with the demo!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateKey](#)
  - [DescribeKey](#)
  - [DisableKey](#)
  - [EnableKey](#)
  - [GenerateDataKey](#)
  - [ListKeys](#)
  - [ScheduleKeyDeletion](#)

## Code examples for Amazon Keyspaces using AWS SDKs

The following code examples show how to use Amazon Keyspaces (for Apache Cassandra) with an AWS software development kit (SDK).

### Get started

#### Hello Amazon Keyspaces

The following code example shows how to get started using Amazon Keyspaces (for Apache Cassandra).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import boto3

def hello_keyspaces(keyspaces_client):
    """
        Use the AWS SDK for Python (Boto3) to create an Amazon Keyspaces (for Apache
        Cassandra)
        client and list the keyspaces in your account.
        This example uses the default settings specified in your shared credentials
        and config files.

        :param keyspaces_client: A Boto3 Amazon Keyspaces Client object. This object
        wraps
                            the low-level Amazon Keyspaces service API.
    """
    print("Hello, Amazon Keyspaces! Let's list some of your keyspaces:\n")
    for ks in keyspaces_client.list_keyspaces(maxResults=5).get('keyspaces', []):
        print(ks['keyspaceName'])
        print(f"\t{ks['resourceArn']}")

if __name__ == '__main__':
    hello_keyspaces(boto3.client('keyspaces'))
```

- For API details, see [ListKeyspaces](#) in *AWS SDK for Python (Boto3) API Reference*.

#### Code examples

- [Actions for Amazon Keyspaces using AWS SDKs \(p. 1152\)](#)
  - [Create an Amazon Keyspaces keyspace using an AWS SDK \(p. 1153\)](#)
  - [Create an Amazon Keyspaces table using an AWS SDK \(p. 1154\)](#)
  - [Delete an Amazon Keyspaces keyspace using an AWS SDK \(p. 1155\)](#)
  - [Delete an Amazon Keyspaces table using an AWS SDK \(p. 1155\)](#)
  - [Get data about an Amazon Keyspaces keyspace using an AWS SDK \(p. 1156\)](#)
  - [Get data about an Amazon Keyspaces table using an AWS SDK \(p. 1157\)](#)
  - [List Amazon Keyspaces keyspaces using an AWS SDK \(p. 1158\)](#)
  - [List Amazon Keyspaces tables in a keyspace using an AWS SDK \(p. 1159\)](#)
  - [Restore an Amazon Keyspaces table to a point in time using an AWS SDK \(p. 1160\)](#)
  - [Update an Amazon Keyspaces table using an AWS SDK \(p. 1160\)](#)
- [Scenarios for Amazon Keyspaces using AWS SDKs \(p. 1161\)](#)
  - [Get started with Amazon Keyspaces keyspaces and tables using an AWS SDK \(p. 1161\)](#)

## Actions for Amazon Keyspaces using AWS SDKs

The following code examples show how to use Amazon Keyspaces (for Apache Cassandra) with AWS SDKs. Each example calls an individual service function.

#### Examples

- [Create an Amazon Keyspaces keyspace using an AWS SDK \(p. 1153\)](#)
- [Create an Amazon Keyspaces table using an AWS SDK \(p. 1154\)](#)
- [Delete an Amazon Keyspaces keyspace using an AWS SDK \(p. 1155\)](#)
- [Delete an Amazon Keyspaces table using an AWS SDK \(p. 1155\)](#)
- [Get data about an Amazon Keyspaces keyspace using an AWS SDK \(p. 1156\)](#)
- [Get data about an Amazon Keyspaces table using an AWS SDK \(p. 1157\)](#)
- [List Amazon Keyspaces keyspaces using an AWS SDK \(p. 1158\)](#)
- [List Amazon Keyspaces tables in a keyspace using an AWS SDK \(p. 1159\)](#)
- [Restore an Amazon Keyspaces table to a point in time using an AWS SDK \(p. 1160\)](#)
- [Update an Amazon Keyspaces table using an AWS SDK \(p. 1160\)](#)

## Create an Amazon Keyspaces keyspace using an AWS SDK

The following code example shows how to create an Amazon Keyspaces keyspace.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:  
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table  
    actions."""  
    def __init__(self, keyspaces_client):  
        """  
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.  
        """  
        self.keyspace_client = keyspaces_client  
        self.ks_name = None  
        self.ks_arn = None  
        self.table_name = None  
  
    @classmethod  
    def from_client(cls):  
        keyspaces_client = boto3.client('keyspaces')  
        return cls(keyspaces_client)  
  
    def create_keyspace(self, name):  
        """  
        Creates a keyspace.  
  
        :param name: The name to give the keyspace.  
        :return: The Amazon Resource Name (ARN) of the new keyspace.  
        """  
        try:  
            response = self.keyspace_client.create_keyspace(keyspaceName=name)  
            self.ks_name = name  
            self.ks_arn = response['resourceArn']  
        except ClientError as err:  
            logger.error(  
                "Couldn't create %s. Here's why: %s: %s", name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return self.ks_arn
```

- For API details, see [CreateKeyspace](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an Amazon Keyspaces table using an AWS SDK

The following code example shows how to create an Amazon Keyspaces table.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:  
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table  
actions."""  
    def __init__(self, keyspaces_client):  
        """  
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.  
        """  
        self.keyspace_client = keyspaces_client  
        self.ks_name = None  
        self.ks_arn = None  
        self.table_name = None  
  
    @classmethod  
    def from_client(cls):  
        keyspaces_client = boto3.client('keyspaces')  
        return cls(keyspaces_client)  
  
    def create_table(self, table_name):  
        """  
        Creates a table in the keyspace.  
        The table is created with a schema for storing movie data  
        and has point-in-time recovery enabled.  
  
        :param table_name: The name to give the table.  
        :return: The ARN of the new table.  
        """  
        try:  
            response = self.keyspace_client.create_table(  
                keyspaceName=self.ks_name, tableName=table_name,  
                schemaDefinition=[  
                    'allColumns': [  
                        {'name': 'title', 'type': 'text'},  
                        {'name': 'year', 'type': 'int'},  
                        {'name': 'release_date', 'type': 'timestamp'},  
                        {'name': 'plot', 'type': 'text'},  
                    ],  
                    'partitionKeys': [{'name': 'year'}, {'name': 'title'}]  
                ],  
                pointInTimeRecovery={'status': 'ENABLED'})  
        except ClientError as err:  
            logger.error(  
                "Couldn't create table %s. Here's why: %s: %s", table_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return response['resourceArn']
```

- For API details, see [CreateTable](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an Amazon Keyspaces keyspace using an AWS SDK

The following code example shows how to delete an Amazon Keyspaces keyspace.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:  
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table  
    actions."""  
    def __init__(self, keyspaces_client):  
        """  
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.  
        """  
        self.keyspace_client = keyspaces_client  
        self.ks_name = None  
        self.ks_arn = None  
        self.table_name = None  
  
    @classmethod  
    def from_client(cls):  
        keyspace_client = boto3.client('keyspaces')  
        return cls(keyspace_client)  
  
    def delete_keyspace(self):  
        """  
        Deletes the keyspace.  
        """  
        try:  
            self.keyspace_client.delete_keyspace(keyspaceName=self.ks_name)  
            self.ks_name = None  
        except ClientError as err:  
            logger.error(  
                "Couldn't delete keyspace %s. Here's why: %s: %s", self.ks_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise
```

- For API details, see [DeleteKeyspace](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an Amazon Keyspaces table using an AWS SDK

The following code example shows how to delete an Amazon Keyspaces table.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:  
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table  
actions."""  
    def __init__(self, keyspaces_client):  
        """  
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.  
        """  
        self.keyspace_client = keyspaces_client  
        self.ks_name = None  
        self.ks_arn = None  
        self.table_name = None  
  
    @classmethod  
    def from_client(cls):  
        keyspace_client = boto3.client('keyspaces')  
        return cls(keyspace_client)  
  
    def delete_table(self):  
        """  
        Deletes the table from the keyspace.  
        """  
        try:  
            self.keyspace_client.delete_table(  
                keyspaceName=self.ks_name, tableName=self.table_name)  
            self.table_name = None  
        except ClientError as err:  
            logger.error(  
                "Couldn't delete table %s. Here's why: %s: %s", self.table_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
        raise
```

- For API details, see [DeleteTable in AWS SDK for Python \(Boto3\) API Reference](#).

## Get data about an Amazon Keyspaces keyspace using an AWS SDK

The following code example shows how to get data about an Amazon Keyspaces keyspace.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:  
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table  
actions."""  
    def __init__(self, keyspace_client):  
        """  
        :param keyspace_client: A Boto3 Amazon Keyspaces client.  
        """  
        self.keyspace_client = keyspace_client  
        self.ks_name = None  
        self.ks_arn = None  
        self.table_name = None  
  
    @classmethod
```

```

def from_client(cls):
    keyspaces_client = boto3.client('keyspaces')
    return cls(keyspaces_client)

def exists_keyspace(self, name):
    """
    Checks whether a keyspace exists.

    :param name: The name of the keyspace to look up.
    :return: True when the keyspace exists. Otherwise, False.
    """
    try:
        response = self.keyspaces_client.get_keyspace(keyspaceName=name)
        self.ks_name = response['keyspaceName']
        self.ks_arn = response['resourceArn']
        exists = True
    except ClientError as err:
        if err.response['Error']['Code'] == 'ResourceNotFoundException':
            logger.info("Keyspace %s does not exist.", name)
            exists = False
        else:
            logger.error(
                "Couldn't verify %s exists. Here's why: %s: %s",
                name,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
    return exists

```

- For API details, see [GetKeyspace](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get data about an Amazon Keyspaces table using an AWS SDK

The following code example shows how to get data about an Amazon Keyspaces table.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class KeyspaceWrapper:
    """
    Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions.
    """
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client('keyspaces')
        return cls(keyspaces_client)

    def get_table(self, table_name):
        """
        """

```

```
Gets data about a table in the keyspace.

:param table_name: The name of the table to look up.
:return: Data about the table.
"""
try:
    response = self.keyspaces_client.get_table(
        keyspaceName=self.ks_name, tableName=table_name)
    self.table_name = table_name
except ClientError as err:
    if err.response['Error']['Code'] == 'ResourceNotFoundException':
        logger.info("Table %s does not exist.", table_name)
        self.table_name = None
        response = None
    else:
        logger.error(
            "Couldn't verify %s exists. Here's why: %s: %s",
            table_name,
            err.response['Error']['Code'], err.response['Error']
            ['Message'])
        raise
return response
```

- For API details, see [GetTable](#) in *AWS SDK for Python (Boto3) API Reference*.

## List Amazon Keyspaces keyspaces using an AWS SDK

The following code example shows how to list Amazon Keyspaces keyspaces.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client('keyspaces')
        return cls(keyspaces_client)

    def list_keyspaces(self, limit):
        """
        Lists the keyspaces in your account.

        :param limit: The maximum number of keyspaces to list.
        """
        try:
            ksPaginator = self.keyspaces_client.get_paginator('list_keyspaces')
            # ...
        except ClientError as err:
            logger.error(err)
            raise
```

```
        for page in ksPaginator.paginate(PaginationConfig={'MaxItems': limit}):
            for ks in page['keyspaces']:
                print(ks['keyspaceName'])
                print(f"\t{ks['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list keyspaces. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

- For API details, see [ListKeyspaces in AWS SDK for Python \(Boto3\) API Reference](#).

## List Amazon Keyspaces tables in a keyspace using an AWS SDK

The following code example shows how to list Amazon Keyspaces tables in a keyspace.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client('keyspaces')
        return cls(keyspaces_client)

    def list_tables(self):
        """
        Lists the tables in the keyspace.
        """
        try:
            tablePaginator = self.keyspaces_client.getPaginator('list_tables')
            for page in tablePaginator.paginate(keyspaceName=self.ks_name):
                for table in page['tables']:
                    print(table['tableName'])
                    print(f"\t{table['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list tables in keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

- For API details, see [ListTables in AWS SDK for Python \(Boto3\) API Reference](#).

## Restore an Amazon Keyspaces table to a point in time using an AWS SDK

The following code example shows how to restore an Amazon Keyspaces table to a point in time.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:  
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table  
    actions."""  
    def __init__(self, keyspaces_client):  
        """  
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.  
        """  
        self.keyspace_client = keyspaces_client  
        self.ks_name = None  
        self.ks_arn = None  
        self.table_name = None  
  
    @classmethod  
    def from_client(cls):  
        keyspace_client = boto3.client('keyspaces')  
        return cls(keyspace_client)  
  
    def restore_table(self, restore_timestamp):  
        """  
        Restores the table to a previous point in time. The table is restored  
        to a new table in the same keyspace.  
  
        :param restore_timestamp: The point in time to restore the table. This time  
            must be in UTC format.  
        :return: The name of the restored table.  
        """  
        try:  
            restored_table_name = f"{self.table_name}_restored"  
            self.keyspace_client.restore_table(  
                sourceKeyspaceName=self.ks_name, sourceTableName=self.table_name,  
                targetKeyspaceName=self.ks_name,  
                targetTableName=restored_table_name,  
                restoreTimestamp=restore_timestamp)  
        except ClientError as err:  
            logger.error(  
                "Couldn't restore table %s. Here's why: %s: %s", restore_timestamp,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return restored_table_name
```

- For API details, see [RestoreTable](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update an Amazon Keyspaces table using an AWS SDK

The following code example shows how to update an Amazon Keyspaces table.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspace_client = keyspace_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspace_client = boto3.client('keyspaces')
        return cls(keyspace_client)

    def update_table(self):
        """
        Updates the schema of the table.

        This example updates a table of movie data by adding a new column
        that tracks whether the movie has been watched.
        """
        try:
            self.keyspace_client.update_table(
                keyspaceName=self.ks_name, tableName=self.table_name,
                addColumns=[{'name': 'watched', 'type': 'boolean'}])
        except ClientError as err:
            logger.error(
                "Couldn't update table %s. Here's why: %s: %s",
                self.table_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

- For API details, see [UpdateTable in AWS SDK for Python \(Boto3\) API Reference](#).

## Scenarios for Amazon Keyspaces using AWS SDKs

The following code examples show how to use Amazon Keyspaces (for Apache Cassandra) with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Get started with Amazon Keyspaces keyspaces and tables using an AWS SDK \(p. 1161\)](#)

## Get started with Amazon Keyspaces keyspaces and tables using an AWS SDK

The following code example shows how to:

- Create a keyspace.
- Create a table in the keyspace. The table is configured with a schema to hold movie data and has point-in-time recovery enabled.
- Connect to the keyspace with a connection secured by TLS and authenticated with Signature V4 (SigV4).
- Query the table by adding, retrieving, and updating movie data.
- Update the table by adding a column to track watched movies.
- Restore the table to a previous point in time.
- Delete the table and keyspace.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
class KeyspaceScenario:  
    """Runs an interactive scenario that shows how to get started using Amazon  
    Keyspaces."""  
    def __init__(self, ks_wrapper):  
        """  
        :param ks_wrapper: An object that wraps Amazon Keyspace actions.  
        """  
        self.ks_wrapper = ks_wrapper  
  
    @demo_func  
    def create_keyspace(self):  
        """  
        1. Creates a keyspace.  
        2. Lists up to 10 keyspaces in your account.  
        """  
        print("Let's create a keyspace.")  
        ks_name = q.ask(  
            "Enter a name for your new keyspace.\nThe name can contain only  
            letters, "  
            "numbers and underscores: ", q.non_empty)  
        if self.ks_wrapper.exists_keyspace(ks_name):  
            print(f"A keyspace named {ks_name} exists.")  
        else:  
            ks_arn = self.ks_wrapper.create_keyspace(ks_name)  
            ks_exists = False  
            while not ks_exists:  
                wait(3)  
                ks_exists = self.ks_wrapper.exists_keyspace(ks_name)  
            print(f"Created a new keyspace.\n\t{ks_arn}.")  
        print("The first 10 keyspaces in your account are:\n")  
        self.ks_wrapper.list_keyspaces(10)  
  
    @demo_func  
    def create_table(self):  
        """  
        1. Creates a table in the keyspace. The table is configured with a schema  
        to hold  
            movie data and has point-in-time recovery enabled.  
        2. Waits for the table to be in an active state.  
        3. Displays schema information for the table.  
        """
```

```

4. Lists tables in the keyspace.
"""
print("Let's create a table for movies in your keyspace.")
table_name = q.ask("Enter a name for your table: ", q.non_empty)
table = self.ks_wrapper.get_table(table_name)
if table is not None:
    print(f"A table named {table_name} already exists in keyspace "
          f"{self.ks_wrapper.ks_name}.")
else:
    table_arn = self.ks_wrapper.create_table(table_name)
    print(f"Created table {table_name}:\\n\\t{table_arn}")
    table = {'status': None}
    print("Waiting for your table to be ready...")
    while table['status'] != 'ACTIVE':
        wait(5)
        table = self.ks_wrapper.get_table(table_name)
    print(f"Your table is {table['status']}. Its schema is:")
    pp(table['schemaDefinition'])
    print("\nThe tables in your keyspace are:\\n")
    self.ks_wrapper.list_tables()

@demo_func
def ensure_tls_cert(self):
    """
    Ensures you have a TLS certificate available to use to secure the
    connection
    to the keyspace. This function downloads a default certificate or lets you
    specify your own.
    """
    print("To connect to your keyspace, you must have a TLS certificate.")
    print("Checking for TLS certificate...")
    cert_path = os.path.join(os.path.dirname(__file__),
                            QueryManager.DEFAULT_CERT_FILE)
    if not os.path.exists(cert_path):
        if q.ask(f"Do you want to download one from {QueryManager.CERT_URL}?
(y/n) ",
                q.is_yesno):
            cert = requests.get(QueryManager.CERT_URL).text
            with open(cert_path, 'w') as cert_file:
                cert_file.write(cert)
        else:
            cert_path = q.ask(
                "Enter the full path to the TLS certificate you want to use: ",
                q.non_empty)
            print(f"Certificate {cert_path} will be used to secure the connection to
your keyspace.")
    return cert_path

@demo_func
def query_table(self, qm):
    """
    1. Adds movies to the table from a sample movie data file.
    2. Gets a list of movies from the table and lets you select one.
    3. Displays more information about the selected movie.
    """
    qm.add_movies(self.ks_wrapper.table_name, '../../../../../resources/sample_files/
movies.json')
    movies = qm.get_movies(self.ks_wrapper.table_name)
    print(f"Added {len(movies)} movies to the table:")
    sel = q.choose("Pick one to learn more about it: ", [m.title for m in
movies])
    movie_choice = qm.get_movie(self.ks_wrapper.table_name, movies[sel].title,
movies[sel].year)
    print(movie_choice.title)
    print(f"\tReleased: {movie_choice.release_date}")
    print(f"\tPlot: {movie_choice.plot}")

```

```

@demo_func
def update_and_restore_table(self, qm):
    """
    1. Updates the table by adding a column to track watched movies.
    2. Marks some of the movies as watched.
    3. Gets the list of watched movies from the table.
    4. Restores to a movies_restored table at a previous point in time.
    5. Gets the list of movies from the restored table.
    """
    print("Let's add a column to record which movies you've watched.")
    pre_update_timestamp = datetime.utcnow()
    print(f"Recorded the current UTC time of {pre_update_timestamp} so we can
restore the table later.")
    self.ks_wrapper.update_table()
    print("Waiting for your table to update...")
    table = {'status': 'UPDATING'}
    while table['status'] != 'ACTIVE':
        wait(5)
        table = self.ks_wrapper.get_table(self.ks_wrapper.table_name)
    print("Column 'watched' added to table.")
    q.ask("Let's mark some of the movies as watched. Press Enter when you're
ready.\n")
    movies = qm.get_movies(self.ks_wrapper.table_name)
    for movie in movies[:10]:
        qm.watched_movie(self.ks_wrapper.table_name, movie.title, movie.year)
        print(f"Marked {movie.title} as watched.")
    movies = qm.get_movies(self.ks_wrapper.table_name, watched=True)
    print('*'*88)
    print("The watched movies in our table are:\n")
    for movie in movies:
        print(movie.title)
    print('*'*88)
    if q.ask(
            "Do you want to restore the table to the way it was before all of
these\n"
            "updates? Keep in mind, this can take up to 20 minutes. (y/n) ",
        q.is_yesno):
        starting_table_name = self.ks_wrapper.table_name
        table_name_restored =
        self.ks_wrapper.restore_table(pre_update_timestamp)
        table = {'status': 'RESTORING'}
        while table['status'] != 'ACTIVE':
            wait(10)
            table = self.ks_wrapper.get_table(table_name_restored)
        print(f"Restored {starting_table_name} to {table_name_restored} "
              f"at a point in time of {pre_update_timestamp}.")
        movies = qm.get_movies(table_name_restored)
        print("Now the movies in our table are:")
        for movie in movies:
            print(movie.title)

    def cleanup(self):
        """
        1. Deletes the table and waits for it to be removed.
        2. Deletes the keyspace.
        """
        if q.ask(f"Do you want to delete your {self.ks_wrapper.table_name} table
and "
                f"\n{self.ks_wrapper.ks_name} keyspace? (y/n) ", q.is_yesno):
            table_name = self.ks_wrapper.table_name
            self.ks_wrapper.delete_table()
            table = self.ks_wrapper.get_table(table_name)
            print("Waiting for the table to be deleted.")
            while table is not None:
                wait(5)

```

```

        table = self.ks_wrapper.get_table(table_name)
        print("Table deleted.")
        self.ks_wrapper.delete_keyspace()
        print("Keyspace deleted. If you chose to restore your table during the
"
               "demo, the original table is also deleted.")

    def run_scenario(self):
        logging.basicConfig(level=logging.INFO, format='%(levelname)s:
%(message)s')

        print('*'*88)
        print("Welcome to the Amazon Keyspaces (for Apache Cassandra) demo.")
        print('*'*88)

        self.create_keyspace()
        self.create_table()
        cert_file_path = self.ensure_tls_cert()
        # Use a context manager to ensure the connection to the keyspace is closed.
        with QueryManager(
            cert_file_path, boto3.DEFAULT_SESSION, self.ks_wrapper.ks_name) as
        qm:
            self.query_table(qm)
            self.update_and_restore_table(qm)
            self.cleanup()

        print("\nThanks for watching!")
        print('*'*88)

if __name__ == '__main__':
    try:
        scenario = KeyspaceScenario(KeyspaceWrapper.from_client())
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")

```

Define a class that wraps keyspace and table actions.

```

class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
actions."""
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client('keyspaces')
        return cls(keyspaces_client)

    def create_keyspace(self, name):
        """
        Creates a keyspace.

        :param name: The name to give the keyspace.
        :return: The Amazon Resource Name (ARN) of the new keyspace.
        """
        try:

```

```
        response = self.keyspaces_client.create_keyspace(keyspaceName=name)
        self.ks_name = name
        self.ks_arn = response['resourceArn']
    except ClientError as err:
        logger.error(
            "Couldn't create %s. Here's why: %s: %s",
            name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return self.ks_arn

def exists_keyspace(self, name):
    """
    Checks whether a keyspace exists.

    :param name: The name of the keyspace to look up.
    :return: True when the keyspace exists. Otherwise, False.
    """
    try:
        response = self.keyspaces_client.get_keyspace(keyspaceName=name)
        self.ks_name = response['keyspaceName']
        self.ks_arn = response['resourceArn']
        exists = True
    except ClientError as err:
        if err.response['Error']['Code'] == 'ResourceNotFoundException':
            logger.info("Keyspace %s does not exist.", name)
            exists = False
        else:
            logger.error(
                "Couldn't verify %s exists. Here's why: %s: %s",
                name,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
    raise
    return exists

def list_keyspaces(self, limit):
    """
    Lists the keyspaces in your account.

    :param limit: The maximum number of keyspaces to list.
    """
    try:
        ksPaginator = self.keyspaces_client.getPaginator('list_keyspaces')
        for page in ksPaginator.paginate(PaginationConfig={'MaxItems': limit}):
            for ks in page['keyspaces']:
                print(ks['keyspaceName'])
                print(f"\t{ks['resourceArn']}")
    except ClientError as err:
        logger.error(
            "Couldn't list keyspaces. Here's why: %s: %s",
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise

def create_table(self, table_name):
    """
    Creates a table in the keyspace.
    The table is created with a schema for storing movie data
    and has point-in-time recovery enabled.

    :param table_name: The name to give the table.
    :return: The ARN of the new table.
    """
    try:
        response = self.keyspaces_client.create_table(
            keySpaceName=self.ks_name, tableName=table_name,
```

```

        schemaDefinition={
            'allColumns': [
                {'name': 'title', 'type': 'text'},
                {'name': 'year', 'type': 'int'},
                {'name': 'release_date', 'type': 'timestamp'},
                {'name': 'plot', 'type': 'text'},
            ],
            'partitionKeys': [{'name': 'year'}, {'name': 'title'}]
        },
        pointInTimeRecovery={'status': 'ENABLED'})
    except ClientError as err:
        logger.error(
            "Couldn't create table %s. Here's why: %s: %s",
            table_name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return response['resourceArn']

def get_table(self, table_name):
    """
    Gets data about a table in the keyspace.

    :param table_name: The name of the table to look up.
    :return: Data about the table.
    """
    try:
        response = self.keyspaces_client.get_table(
            keyspaceName=self.ks_name, tableName=table_name)
        self.table_name = table_name
    except ClientError as err:
        if err.response['Error']['Code'] == 'ResourceNotFoundException':
            logger.info("Table %s does not exist.", table_name)
            self.table_name = None
            response = None
        else:
            logger.error(
                "Couldn't verify %s exists. Here's why: %s: %s",
                table_name,
                err.response['Error']['Code'],
                err.response['Error']
            ['Message']))
            raise
    return response

def list_tables(self):
    """
    Lists the tables in the keyspace.
    """
    try:
        tablePaginator = self.keyspaces_client.getPaginator('list_tables')
        for page in tablePaginator.paginate(keyspaceName=self.ks_name):
            for table in page['tables']:
                print(table['tableName'])
                print(f"\t{table['resourceArn']}")
    except ClientError as err:
        logger.error(
            "Couldn't list tables in keyspace %s. Here's why: %s: %s",
            self.ks_name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise

def update_table(self):
    """
    Updates the schema of the table.

    This example updates a table of movie data by adding a new column
    that tracks whether the movie has been watched.
    """

```

```

try:
    self.keyspaces_client.update_table(
        keyspaceName=self.ks_name, tableName=self.table_name,
        addColumns=[{'name': 'watched', 'type': 'boolean'}])
except ClientError as err:
    logger.error(
        "Couldn't update table %s. Here's why: %s: %s", self.table_name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise

def restore_table(self, restore_timestamp):
    """
    Restores the table to a previous point in time. The table is restored
    to a new table in the same keyspace.

    :param restore_timestamp: The point in time to restore the table. This time
                             must be in UTC format.
    :return: The name of the restored table.
    """
    try:
        restored_table_name = f"{self.table_name}_restored"
        self.keyspaces_client.restore_table(
            sourceKeyspaceName=self.ks_name, sourceTableName=self.table_name,
            targetKeyspaceName=self.ks_name,
            targetTableName=restored_table_name,
            restoreTimestamp=restore_timestamp)
    except ClientError as err:
        logger.error(
            "Couldn't restore table %s. Here's why: %s: %s", restore_timestamp,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return restored_table_name

def delete_table(self):
    """
    Deletes the table from the keyspace.
    """
    try:
        self.keyspaces_client.delete_table(
            keyspaceName=self.ks_name, tableName=self.table_name)
        self.table_name = None
    except ClientError as err:
        logger.error(
            "Couldn't delete table %s. Here's why: %s: %s", self.table_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def delete_keyspace(self):
    """
    Deletes the keyspace.
    """
    try:
        self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
        self.ks_name = None
    except ClientError as err:
        logger.error(
            "Couldn't delete keyspace %s. Here's why: %s: %s", self.ks_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

```

Define a class that creates a TLS connection to a keyspace, authenticates with SigV4, and sends CQL queries to a table in the keyspace.

```
class QueryManager:  
    """  
        Manages queries to an Amazon Keyspaces (for Apache Cassandra) keyspace.  
        Queries are secured by TLS and authenticated by using the Signature V4 (SigV4)  
        AWS signing protocol. This is more secure than sending username and password  
        with a plain-text authentication provider.  
  
        This example downloads a default certificate to secure TLS, or lets you specify  
        your own.  
  
        This example uses a table of movie data to demonstrate basic queries.  
    """  
    DEFAULT_CERT_FILE = 'sf-class2-root.crt'  
    CERT_URL = f'https://certs.secureserver.net/repository/sf-class2-root.crt'  
  
    def __init__(self, cert_file_path, boto_session, keyspace_name):  
        """  
            :param cert_file_path: The path and file name of the certificate used for  
            TLS.  
            :param boto_session: A Boto3 session. This is used to acquire your AWS  
            credentials.  
            :param keyspace_name: The name of the keyspace to connect.  
        """  
        self.cert_file_path = cert_file_path  
        self.boto_session = boto_session  
        self.ks_name = keyspace_name  
        self.cluster = None  
        self.session = None  
  
    def __enter__(self):  
        """  
            Creates a session connection to the keyspace that is secured by TLS and  
            authenticated by SigV4.  
        """  
        ssl_context = SSLContext(PROTOCOL_TLSv1_2)  
        ssl_context.load_verify_locations(self.cert_file_path)  
        ssl_context.verify_mode = CERT_REQUIRED  
        auth_provider = SigV4AuthProvider(self.boto_session)  
        contact_point = f"cassandra.{self.boto_session.region_name}.amazonaws.com"  
        exec_profile = ExecutionProfile(  
            consistency_level=ConsistencyLevel.LOCAL_QUORUM,  
            load_balancing_policy=DCAwareRoundRobinPolicy())  
        self.cluster = Cluster(  
            [contact_point], ssl_context=ssl_context, auth_provider=auth_provider,  
            port=9142, execution_profiles={EXEC_PROFILE_DEFAULT: exec_profile},  
            protocol_version=4)  
        self.cluster.__enter__()  
        self.session = self.cluster.connect(self.ks_name)  
        return self  
  
    def __exit__(self, *args):  
        """  
            Exits the cluster. This shuts down all existing session connections.  
        """  
        self.cluster.__exit__(*args)  
  
    def add_movies(self, table_name, movie_file_path):  
        """  
            Gets movies from a JSON file and adds them to a table in the keyspace.  
  
            :param table_name: The name of the table.  
            :param movie_file_path: The path and file name of a JSON file that contains  
            movie data.  
        """  
        with open(movie_file_path, 'r') as movie_file:
```

```

movies = json.loads(movie_file.read())
stmt = self.session.prepare(
    f"INSERT INTO {table_name} (year, title, release_date, plot) VALUES
(?, ?, ?, ?);")
for movie in movies[:20]:
    self.session.execute(stmt, parameters=[
        movie['year'], movie['title'],
        date.fromisoformat(movie['info']['release_date'].partition('T')
[0]),
        movie['info']['plot']])

def get_movies(self, table_name, watched=None):
    """
    Gets the title and year of the full list of movies from the table.

    :param table_name: The name of the movie table.
    :param watched: When specified, the returned list of movies is filtered to
                    either movies that have been watched or movies that have
                    not
                    been watched. Otherwise, all movies are returned.
    :return: A list of movies in the table.
    """
    if watched is None:
        stmt = SimpleStatement(f"SELECT title, year FROM {table_name}")
        params = None
    else:
        stmt = SimpleStatement(
            f"SELECT title, year FROM {table_name} WHERE watched = %s ALLOW
FILTERING")
        params = [watched]
    return self.session.execute(stmt, parameters=params).all()

def get_movie(self, table_name, title, year):
    """
    Gets a single movie from the table, by title and year.

    :param table_name: The name of the movie table.
    :param title: The title of the movie.
    :param year: The year of the movie's release.
    :return: The requested movie.
    """
    return self.session.execute(
        SimpleStatement(f"SELECT * FROM {table_name} WHERE title = %s AND year
= %s"),
        parameters=[title, year]).one()

def watched_movie(self, table_name, title, year):
    """
    Updates a movie as having been watched.

    :param table_name: The name of the movie table.
    :param title: The title of the movie.
    :param year: The year of the movie's release.
    """
    self.session.execute(
        SimpleStatement(f"UPDATE {table_name} SET watched=true WHERE title = %s
AND year = %s"),
        parameters=[title, year])

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.

- [CreateKeyspace](#)
- [CreateTable](#)
- [DeleteKeyspace](#)

- [DeleteTable](#)
- [GetKeyspace](#)
- [GetTable](#)
- [ListKeyspaces](#)
- [ListTables](#)
- [RestoreTable](#)
- [UpdateTable](#)

## Code examples for Kinesis using AWS SDKs

The following code examples show how to use Amazon Kinesis with an AWS software development kit (SDK).

### Code examples

- [Actions for Kinesis using AWS SDKs \(p. 1171\)](#)
  - [Add tags to a Kinesis stream using an AWS SDK \(p. 1172\)](#)
  - [Create a Kinesis stream using an AWS SDK \(p. 1173\)](#)
  - [Delete a Kinesis stream using an AWS SDK \(p. 1176\)](#)
  - [Deregister a consumer from a Kinesis stream using an AWS SDK \(p. 1179\)](#)
  - [Describe a Kinesis stream using an AWS SDK \(p. 1180\)](#)
  - [Get data in batches from a Kinesis stream using an AWS SDK \(p. 1182\)](#)
  - [List Kinesis streams using an AWS SDK \(p. 1185\)](#)
  - [List the tags associated with a Kinesis stream using an AWS SDK \(p. 1187\)](#)
  - [List the consumers of a Kinesis stream using an AWS SDK \(p. 1188\)](#)
  - [Put data into a Kinesis stream using an AWS SDK \(p. 1189\)](#)
  - [Register a consumer to a Kinesis stream using an AWS SDK \(p. 1193\)](#)
- [Scenarios for Kinesis using AWS SDKs \(p. 1194\)](#)
  - [Get started with basic Kinesis data stream operations using an AWS SDK \(p. 1194\)](#)

## Actions for Kinesis using AWS SDKs

The following code examples show how to use Amazon Kinesis with AWS SDKs. Each example calls an individual service function.

### Examples

- [Add tags to a Kinesis stream using an AWS SDK \(p. 1172\)](#)
- [Create a Kinesis stream using an AWS SDK \(p. 1173\)](#)
- [Delete a Kinesis stream using an AWS SDK \(p. 1176\)](#)
- [Deregister a consumer from a Kinesis stream using an AWS SDK \(p. 1179\)](#)
- [Describe a Kinesis stream using an AWS SDK \(p. 1180\)](#)
- [Get data in batches from a Kinesis stream using an AWS SDK \(p. 1182\)](#)
- [List Kinesis streams using an AWS SDK \(p. 1185\)](#)
- [List the tags associated with a Kinesis stream using an AWS SDK \(p. 1187\)](#)
- [List the consumers of a Kinesis stream using an AWS SDK \(p. 1188\)](#)
- [Put data into a Kinesis stream using an AWS SDK \(p. 1189\)](#)

- Register a consumer to a Kinesis stream using an AWS SDK (p. 1193)

## Add tags to a Kinesis stream using an AWS SDK

The following code example shows how to add tags to a Kinesis stream.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// This example shows how to apply key/value pairs to an Amazon Kinesis
/// stream. The example was created using the AWS SDK for .NET version 3.7
/// and .NET Core 5.0.
/// </summary>
public class TagStream
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();

        string streamName = "AmazonKinesisStream";
        var tags = new Dictionary<string, string>
        {
            { "Project", "Sample Kinesis Project" },
            { "Application", "Sample Kinesis App" },
        };

        var success = await ApplyTagsToStreamAsync(client, streamName, tags);

        if (success)
        {
            Console.WriteLine($"Tags successfully added to {streamName}.");
        }
        else
        {
            Console.WriteLine("Tags were not added to the stream.");
        }
    }

    /// <summary>
    /// Applies the set of tags to the named Kinesis stream.
    /// </summary>
    /// <param name="client">The initialized Kinesis client.</param>
    /// <param name="streamName">The name of the Kinesis stream to which
    /// the tags will be attached.</param>
    /// <param name="tags">A sictionary containing key/value pairs which
    /// will be used to create the Kinesis tags.</param>
    /// <returns>A Boolean value which represents the success or failure
    /// of AddTagsToStreamAsync.</returns>
    public static async Task<bool> ApplyTagsToStreamAsync(
        IAmazonKinesis client,
```

```
        string streamName,
        Dictionary<string, string> tags)
    {
        var request = new AddTagsToStreamRequest
        {
            StreamName = streamName,
            Tags = tags,
        };

        var response = await client.AddTagsToStreamAsync(request);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

- For API details, see [AddTagsToStream](#) in *AWS SDK for .NET API Reference*.

## Create a Kinesis stream using an AWS SDK

The following code examples show how to create a Kinesis stream.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// This example shows how to create a new Amazon Kinesis stream. The
/// example was created using AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class CreateStream
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();

        string streamName = "AmazonKinesisStream";
        int shardCount = 1;

        var success = await CreateNewStreamAsync(client, streamName,
shardCount);
        if (success)
        {
            Console.WriteLine($"The stream, {streamName} successfully
created.");
        }
    }

    /// <summary>
    /// Creates a new Kinesis stream.
    /// </summary>
    /// <param name="client">An initialized Kinesis client.</param>
```

```
    ///<param name="streamName">The name for the new stream.</param>
    ///<param name="shardCount">The number of shards the new stream will
    ///use. The throughput of the stream is a function of the number of
    ///shards; more shards are required for greater provisioned
    ///throughput.</param>
    ///<returns>A Boolean value indicating whether the stream was created.</returns>
    public static async Task<bool> CreateNewStreamAsync(IAmazonKinesis client,
string streamName, int shardCount)
{
    var request = new CreateStreamRequest
    {
        StreamName = streamName,
        ShardCount = shardCount,
    };

    var response = await client.CreateStreamAsync(request);

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [CreateStream](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createStream(KinesisClient kinesisClient, String streamName)
{
    try {
        CreateStreamRequest streamReq = CreateStreamRequest.builder()
            .streamName(streamName)
            .shardCount(1)
            .build();

        kinesisClient.createStream(streamReq);

    } catch (KinesisException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateStream](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""
    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter('stream_exists')

    def create(self, name, wait_until_exists=True):
        """
        Creates a stream.

        :param name: The name of the stream.
        :param wait_until_exists: When True, waits until the service reports that
            the stream exists, then queries for its metadata.
        """
        try:
            self.kinesis_client.create_stream(StreamName=name, ShardCount=1)
            self.name = name
            logger.info("Created stream %s.", name)
            if wait_until_exists:
                logger.info("Waiting until exists.")
                self.stream_exists_waiter.wait(StreamName=name)
                self.describe(name)
        except ClientError:
            logger.exception("Couldn't create stream %s.", name)
            raise
```

- For API details, see [CreateStream in AWS SDK for Python \(Boto3\) API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_stream(client: &Client, stream: &str) -> Result<(), Error> {
    client
        .create_stream()
        .stream_name(stream)
        .shard_count(4)
        .send()
        .await?;

    println!("Created stream");

    Ok(())
}
```

- For API details, see [CreateStream in AWS SDK for Rust API reference](#).

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_kns->createtestream(  
        iv_streamname = iv_stream_name  
        iv_shardcount = iv_shard_count  
    ).  
    MESSAGE 'Stream created' TYPE 'I'.  
    CATCH /aws1/cx_knsinvalidargumentex.  
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.  
    CATCH /aws1/cx_knslimitexceededex .  
        MESSAGE 'The request processing has failed because of a limit exceed  
exception.' TYPE 'E'.  
    CATCH /aws1/cx_knsresourceinuseex .  
        MESSAGE 'The request processing has failed because the resource is in use.'  
    TYPE 'E'.  
ENDTRY.
```

- For API details, see [CreateStream in AWS SDK for SAP ABAP API reference](#).

## Delete a Kinesis stream using an AWS SDK

The following code examples show how to delete a Kinesis stream.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Threading.Tasks;  
using Amazon.Kinesis;  
using Amazon.Kinesis.Model;  
  
/// <summary>  
/// Shows how to delete an Amazon Kinesis stream. The example was created  
/// using the AWS SDK for .NET version 3.7 and .NET Core 5.0.  
/// </summary>  
public class DeleteStream  
{  
    public static async Task Main()  
    {  
        IAmazonKinesis client = new AmazonKinesisClient();  
        string streamName = "AmazonKinesisStream";
```

```
        var success = await DeleteStreamAsync(client, streamName);

        if (success)
        {
            Console.WriteLine($"Stream, {streamName} successfully deleted.");
        }
        else
        {
            Console.WriteLine("Stream not deleted.");
        }
    }

    ///<summary>
    ///</summary>
    ///<param name="client">An initialized Kinesis client object.</param>
    ///<param name="streamName">The name of the string to delete.</param>
    ///<returns>A Boolean value representing the success of the operation.</returns>
public static async Task<bool> DeleteStreamAsync(IAmazonKinesis client,
string streamName)
{
    // If EnforceConsumerDeletion is true, any consumers
    // of this stream will also be deleted. If it is set
    // to false and this stream has any consumers, the
    // call will fail with a ResourceInUseException.
    var request = new DeleteStreamRequest
    {
        StreamName = streamName,
        EnforceConsumerDeletion = true,
    };

    var response = await client.DeleteStreamAsync(request);

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [DeleteStream](#) in [AWS SDK for .NET API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteStream(KinesisClient kinesisClient, String streamName)
{
    try {
        DeleteStreamRequest delStream = DeleteStreamRequest.builder()
            .streamName(streamName)
            .build();

        kinesisClient.deleteStream(delStream);

    } catch (KinesisException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }
```

- For API details, see [DeleteStream](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisStream:  
    """Encapsulates a Kinesis stream."""  
    def __init__(self, kinesis_client):  
        """  
        :param kinesis_client: A Boto3 Kinesis client.  
        """  
        self.kinesis_client = kinesis_client  
        self.name = None  
        self.details = None  
        self.stream_exists_waiter = kinesis_client.get_waiter('stream_exists')  
  
    def delete(self):  
        """  
        Deletes a stream.  
        """  
        try:  
            self.kinesis_client.delete_stream(StreamName=self.name)  
            self._clear()  
            logger.info("Deleted stream %s.", self.name)  
        except ClientError:  
            logger.exception("Couldn't delete stream %s.", self.name)  
            raise
```

- For API details, see [DeleteStream](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn remove_stream(client: &Client, stream: &str) -> Result<(), Error> {  
    client.delete_stream().stream_name(stream).send().await?;  
  
    println!("Deleted stream.");  
  
    Ok(())  
}
```

- For API details, see [DeleteStream](#) in *AWS SDK for Rust API reference*.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_kns->deletestream(  
        iv_streamname = iv_stream_name  
    ).  
    MESSAGE 'Stream deleted' TYPE 'I'.  
    CATCH /aws1/cx_knslimitexceededex .  
        MESSAGE 'The request processing has failed because of a limit exceed  
exception.' TYPE 'E'.  
        CATCH /aws1/cx_knsresourceinuseex .  
            MESSAGE 'The request processing has failed because the resource is in use.'  
TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteStream](#) in *AWS SDK for SAP ABAP API reference*.

## Deregister a consumer from a Kinesis stream using an AWS SDK

The following code example shows how to deregister a consumer from a Kinesis stream.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Threading.Tasks;  
using Amazon.Kinesis;  
using Amazon.Kinesis.Model;  
  
/// <summary>  
/// Shows how to deregister a consumer from an Amazon Kinesis stream. The  
/// example was written using the AWS SDK for .NET 3.7 and .NET Core 5.0.  
/// </summary>  
public class DeregisterConsumer  
{  
    public static async Task Main(string[] args)  
    {  
        IAmazonKinesis client = new AmazonKinesisClient();
```

```
        string streamARN = "arn:aws:kinesis:us-west-2:000000000000:stream/"  
AmazonKinesisStream";  
        string consumerName = "CONSUMER_NAME";  
        string consumerARN = "arn:aws:kinesis:us-west-2:000000000000:stream/"  
AmazonKinesisStream/consumer/CONSUMER_NAME:000000000000";  
  
        var success = await DeregisterConsumerAsync(client, streamARN,  
consumerARN, consumerName);  
  
        if (success)  
        {  
            Console.WriteLine($"{consumerName} successfully deregistered.");  
        }  
        else  
        {  
            Console.WriteLine($"{consumerName} was not successfully  
deregistered.");  
        }  
    }  
  
    /// <summary>  
    /// Deregisters a consumer from a Kinesis stream.  
    /// </summary>  
    /// <param name="client">An initialized Kinesis client object.</param>  
    /// <param name="streamARN">The ARN of a Kinesis stream.</param>  
    /// <param name="consumerARN">The ARN of the consumer.</param>  
    /// <param name="consumerName">The name of the consumer.</param>  
    /// <returns>A Boolean value representing the success of the operation.</returns>  
    public static async Task<bool> DeregisterConsumerAsync(  
        IAmazonKinesis client,  
        string streamARN,  
        string consumerARN,  
        string consumerName)  
    {  
        var request = new DeregisterStreamConsumerRequest  
        {  
            StreamARN = streamARN,  
            ConsumerARN = consumerARN,  
            ConsumerName = consumerName,  
        };  
  
        var response = await client.DeregisterStreamConsumerAsync(request);  
  
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;  
    }  
}
```

- For API details, see [DeregisterStreamConsumer](#) in [AWS SDK for .NET API Reference](#).

## Describe a Kinesis stream using an AWS SDK

The following code examples show how to describe a Kinesis stream.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""
    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter('stream_exists')

    def describe(self, name):
        """
        Gets metadata about a stream.

        :param name: The name of the stream.
        :return: Metadata about the stream.
        """
        try:
            response = self.kinesis_client.describe_stream(StreamName=name)
            self.name = name
            self.details = response['StreamDescription']
            logger.info("Got stream %s.", name)
        except ClientError:
            logger.exception("Couldn't get %s.", name)
            raise
        else:
            return self.details
```

- For API details, see [DescribeStream](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_stream(client: &Client, stream: &str) -> Result<(), Error> {
    let resp = client.describe_stream().stream_name(stream).send().await?;

    let desc = resp.stream_description.unwrap();

    println!("Stream description:");
    println!("  Name:          {}", desc.stream_name.unwrap());
    println!("  Status:       {}", desc.stream_status.unwrap());
    println!("  Open shards:  {}", desc.shards.unwrap().len());
    println!(
        "  Retention (hours): {}",
        desc.retention_period_hours.unwrap()
    );
    println!("  Encryption:   {}", desc.encryption_type.unwrap());

    Ok(())
}
```

- For API details, see [DescribeStream](#) in *AWS SDK for Rust API reference*.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_kns->describestream(  
        iv_streamname = iv_stream_name  
    ).  
    DATA(lt_stream_description) = oo_result->get_streamdescription( ).  
    MESSAGE 'Streams retrieved' TYPE 'I'.  
    CATCH /aws1/cx_knslimitexceededex .  
        MESSAGE 'The request processing has failed because of a limit exceed  
exception.' TYPE 'E'.  
        CATCH /aws1/cx_knsresourcenotfoundex .  
            MESSAGE 'Resource being access is not found.' TYPE 'E'.  
    ENDTRY.
```

- For API details, see [DescribeStream](#) in *AWS SDK for SAP ABAP API reference*.

## Get data in batches from a Kinesis stream using an AWS SDK

The following code examples show how to get data in batches from a Kinesis stream.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getStockTrades(KinesisClient kinesisClient, String  
streamName) {  
  
    String shardIterator;  
    String lastShardId = null;  
  
    // Retrieve the Shards from a Stream  
    DescribeStreamRequest describeStreamRequest =  
DescribeStreamRequest.builder()  
    .streamName(streamName)  
    .build();  
  
    List<Shard> shards = new ArrayList<>();
```

```
DescribeStreamResponse streamRes;
do {
    streamRes = kinesisClient.describeStream(describeStreamRequest);
    shards.addAll(streamRes.streamDescription().shards());

    if (shards.size() > 0) {
        lastShardId = shards.get(shards.size() - 1).shardId();
    }
} while (streamRes.streamDescription().hasMoreShards());

GetShardIteratorRequest itReq = GetShardIteratorRequest.builder()
    .streamName(streamName)
    .shardIteratorType("TRIM_HORIZON")
    .shardId(lastShardId)
    .build();

GetShardIteratorResponse shardIteratorResult =
kinesisClient.getShardIterator(itReq);
shardIterator = shardIteratorResult.shardIterator();

// Continuously read data records from shard.
List<Record> records;

// Create new GetRecordsRequest with existing shardIterator.
// Set maximum records to return to 1000.
GetRecordsRequest recordsRequest = GetRecordsRequest.builder()
    .shardIterator(shardIterator)
    .limit(1000)
    .build();

GetRecordsResponse result = kinesisClient.getRecords(recordsRequest);

// Put result into record list. Result may be empty.
records = result.records();

// Print records
for (Record record : records) {
    SdkBytes byteBuffer = record.data();
    System.out.printf("Seq No: %s - %s%n", record.sequenceNumber(), new
String(byteBuffer.asByteArray()));
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [GetRecords](#)
  - [GetShardIterator](#)

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""
    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
```

```

        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter('stream_exists')

    def get_records(self, max_records):
        """
        Gets records from the stream. This function is a generator that first gets
        a shard iterator for the stream, then uses the shard iterator to get
        records
        in batches from the stream. Each batch of records is yielded back to the
        caller until the specified maximum number of records has been retrieved.

        :param max_records: The maximum number of records to retrieve.
        :return: Yields the current batch of retrieved records.
        """
        try:
            response = self.kinesis_client.get_shard_iterator(
                StreamName=self.name, ShardId=self.details['Shards'][0]['ShardId'],
                ShardIteratorType='LATEST')
            shard_iter = response['ShardIterator']
            record_count = 0
            while record_count < max_records:
                response = self.kinesis_client.get_records(
                    ShardIterator=shard_iter, Limit=10)
                shard_iter = response['NextShardIterator']
                records = response['Records']
                logger.info("Got %s records.", len(records))
                record_count += len(records)
                yield records
        except ClientError:
            logger.exception("Couldn't get records from stream %s.", self.name)
            raise

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [GetRecords](#)
  - [GetShardIterator](#)

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
    oo_result = lo_kns->getrecords(           " oo_result is returned for
testing purpose "
        iv_sharditerator = iv_shard_iterator
    ).
    DATA(lt_records) = oo_result->get_records( ).
    MESSAGE 'Record retrieved' TYPE 'I'.
CATCH /aws1/cx_knsexpirediteratorex .
    MESSAGE 'Iterator expired.' TYPE 'E'.
CATCH /aws1/cx_knsinvalidargumentex .

```

```
MESSAGE 'The specified argument was not valid.' TYPE 'E'.
CATCH /aws1/cx_knskmsaccessdeniedex .
MESSAGE 'You do not have permission to perform this KMS action.' TYPE 'E'.
CATCH /aws1/cx_knskmsdisabledex .
MESSAGE 'KMS key used is disabled' TYPE 'E'.
CATCH /aws1/cx_knskmsinvalidstateex .
MESSAGE 'KMS key used is in an invalid state ' TYPE 'E'.
CATCH /aws1/cx_knskmsnotfoundex .
MESSAGE 'KMS key used is not found' TYPE 'E'.
CATCH /aws1/cx_knskmsoptionrequired .
MESSAGE 'KMS key option is required' TYPE 'E'.
CATCH /aws1/cx_knskmsthrottlingex .
MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
CATCH /aws1/cx_knsprovthruputrexdex .
MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
CATCH /aws1/cx_knsresourcefounddex .
MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [GetRecords](#)
  - [GetShardIterator](#)

## List Kinesis streams using an AWS SDK

The following code examples show how to list information about one or more Kinesis streams.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// Retrieves and displays a list of existing Amazon Kinesis streams. The
/// example uses the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class ListStreams
{
    public static async Task Main(string[] args)
    {
        IAmazonKinesis client = new AmazonKinesisClient();
        var response = await client.ListStreamsAsync(new ListStreamsRequest());

        List<string> streamNames = response.StreamNames;

        if (streamNames.Count > 0)
        {
            streamNames
                .ForEach(s => Console.WriteLine($"Stream name: {s}"));
        }
    }
}
```

```
        }
    else
    {
        Console.WriteLine("No streams were found.");
    }
}
```

- For API details, see [ListStreams in AWS SDK for .NET API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_streams(client: &Client) -> Result<(), Error> {
    let resp = client.list_streams().send().await?;

    println!("Stream names:");

    let streams = resp.stream_names.unwrap_or_default();
    for stream in &streams {
        println!("  {}", stream);
    }

    println!("Found {} stream(s)", streams.len());
    Ok(())
}
```

- For API details, see [ListStreams in AWS SDK for Rust API reference](#).

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_kns->liststreams(          " oo_result is returned for testing
purpose "
                                         "set Limit to specify that a maximum of streams should be returned"
```

```
        iv_limit = iv_limit
    ).
    DATA(lt_streams) = oo_result->get_streamnames( ).
    MESSAGE 'Streams listed' TYPE 'I'.
    CATCH /aws1/cx_knslimitexceededex .
        MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
    ENDTRY.
```

- For API details, see [ListStreams in AWS SDK for SAP ABAP API reference](#).

## List the tags associated with a Kinesis stream using an AWS SDK

The following code example shows how to list the tags associated with a Kinesis stream.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// Shows how to list the tags that have been attached to an Amazon Kinesis
/// stream. The example was created using the AWS SDK for .NET version 3.7
/// and .NET Core 5.0.
/// </summary>
public class ListTags
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();
        string streamName = "AmazonKinesisStream";

        await ListTagsAsync(client, streamName);
    }

    /// <summary>
    /// List the tags attached to a Kinesis stream.
    /// </summary>
    /// <param name="client">An initialized Kinesis client object.</param>
    /// <param name="streamName">The name of the Kinesis stream for which you
    /// wish to display tags.</param>
    public static async Task ListTagsAsync(IAmazonKinesis client, string
streamName)
    {
        var request = new ListTagsForStreamRequest
        {
            StreamName = streamName,
            Limit = 10,
        };

        var response = await client.ListTagsForStreamAsync(request);
```

```
        DisplayTags(response.Tags);

        while (response.HasMoreTags)
        {
            request.ExclusiveStartTagKey = response.Tags[response.Tags.Count - 1].Key;
            response = await client.ListTagsForStreamAsync(request);
        }
    }

    ///<summary>
    ///<summary> Displays the items in a list of Kinesis tags.
    ///</summary>
    ///<param name="tags">A list of the Tag objects to be displayed.</param>
    public static void DisplayTags(List<Tag> tags)
    {
        tags
            .ForEach(t => Console.WriteLine($"Key: {t.Key} Value: {t.Value}"));
    }
}
```

- For API details, see [ListTagsForStream](#) in *AWS SDK for .NET API Reference*.

## List the consumers of a Kinesis stream using an AWS SDK

The following code example shows how to list the consumers of a Kinesis stream.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

    ///<summary>
    ///<summary> List the consumers of an Amazon Kinesis stream. This example was
    ///<summary> created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
    ///</summary>
    public class ListConsumers
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();

            string streamARN = "arn:aws:kinesis:us-east-2:000000000000:stream/
AmazonKinesisStream";
            int maxResults = 10;

            var consumers = await ListConsumersAsync(client, streamARN,
maxResults);

            if (consumers.Count > 0)
```

```
        {
            consumers
                .ForEach(c => Console.WriteLine($"Name: {c.ConsumerName} ARN:
{c.ConsumerARN}"));
        }
        else
        {
            Console.WriteLine("No consumers found.");
        }
    }

/// <summary>
/// Retrieve a list of the consumers for a Kinesis stream.
/// </summary>
/// <param name="client">An initialized Kinesis client object.</param>
/// <param name="streamARN">The ARN of the stream for which we want to
/// retrieve a list of clients.</param>
/// <param name="maxResults">The maximum number of results to return.</param>
/// <returns>A list of Consumer objects.</returns>
public static async Task<List<Consumer>> ListConsumersAsync(IAmazonKinesis
client, string streamARN, int maxResults)
{
    var request = new ListStreamConsumersRequest
    {
        StreamARN = streamARN,
        MaxResults = maxResults,
    };

    var response = await client.ListStreamConsumersAsync(request);

    return response.Consumers;
}
```

- For API details, see [ListStreamConsumers in AWS SDK for .NET API Reference](#).

## Put data into a Kinesis stream using an AWS SDK

The following code examples show how to put data into a Kinesis stream.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void setStockData( KinesisClient kinesisClient, String
streamName) {

    try {
        // Repeatedly send stock trades with a 100 milliseconds wait in
        between.
        StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

        // Put in 50 Records for this example.
        int index = 50;
```

```
        for (int x=0; x<index; x++) {
            StockTrade trade = stockTradeGenerator.getRandomTrade();
            sendStockTrade(trade, kinesisClient, streamName);
            Thread.sleep(100);
        }

    } catch (KinesisException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}

private static void sendStockTrade(StockTrade trade, KinesisClient
kinesisClient,
                                  String streamName) {
    byte[] bytes = trade.toJsonAsBytes();

    // The bytes could be null if there is an issue with the JSON serialization
    // by the Jackson JSON library.
    if (bytes == null) {
        System.out.println("Could not get JSON bytes for stock trade");
        return;
    }

    System.out.println("Putting trade: " + trade);
    PutRecordRequest request = PutRecordRequest.builder()
        .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol as
        // the partition key, explained in the Supplemental Information section below.
        .streamName(streamName)
        .data(SdkBytes.fromByteArray(bytes))
        .build();

    try {
        kinesisClient.putRecord(request);
    } catch (KinesisException e) {
        e.getMessage();
    }
}

private static void validateStream(KinesisClient kinesisClient, String
streamName) {
    try {
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
        .streamName(streamName)
        .build();

        DescribeStreamResponse describeStreamResponse =
kinesisClient.describeStream(describeStreamRequest);

        if(!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
{
            System.err.println("Stream " + streamName + " is not active. Please
wait a few moments and try again.");
            System.exit(1);
        }

    }catch (KinesisException e) {
        System.err.println("Error found while describing the stream " +
streamName);
        System.err.println(e);
        System.exit(1);
    }
}
```

- For API details, see [PutRecord](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisStream:  
    """Encapsulates a Kinesis stream."""  
    def __init__(self, kinesis_client):  
        """  
        :param kinesis_client: A Boto3 Kinesis client.  
        """  
        self.kinesis_client = kinesis_client  
        self.name = None  
        self.details = None  
        self.stream_exists_waiter = kinesis_client.get_waiter('stream_exists')  
  
    def put_record(self, data, partition_key):  
        """  
        Puts data into the stream. The data is formatted as JSON before it is  
        passed  
        to the stream.  
  
        :param data: The data to put in the stream.  
        :param partition_key: The partition key to use for the data.  
        :return: Metadata about the record, including its shard ID and sequence  
        number.  
        """  
        try:  
            response = self.kinesis_client.put_record(  
                StreamName=self.name,  
                Data=json.dumps(data),  
                PartitionKey=partition_key)  
            logger.info("Put record in stream %s.", self.name)  
        except ClientError:  
            logger.exception("Couldn't put record in stream %s.", self.name)  
            raise  
        else:  
            return response
```

- For API details, see [PutRecord](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn add_record(client: &Client, stream: &str, key: &str, data: &str) ->
Result<(), Error> {
    let blob = Blob::new(data);

    client
        .put_record()
        .data(blob)
        .partition_key(key)
        .stream_name(stream)
        .send()
        .await?;

    println!("Put data into stream.");

    Ok(())
}
```

- For API details, see [PutRecord](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_kns->putrecord(                      " oo_result is returned for
testing purpose "
        iv_streamname = iv_stream_name
        iv_data      = iv_data
        iv_partitionkey = iv_partition_key
    ).
    MESSAGE 'Record created' TYPE 'I'.
CATCH /aws1/cx_knsinvalidargumentex .
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
CATCH /aws1/cx_knskmsaccessdeniedex .
    MESSAGE 'You do not have permission to perform this KMS action.' TYPE 'E'.
CATCH /aws1/cx_knskmsdisabledex .
    MESSAGE 'KMS key used is disabled' TYPE 'E'.
CATCH /aws1/cx_knskmsinvalidstateex .
    MESSAGE 'KMS key used is in an invalid state ' TYPE 'E'.
CATCH /aws1/cx_knskmsnotfoundex .
    MESSAGE 'KMS key used is not found' TYPE 'E'.
CATCH /aws1/cx_knskmsoptinrequired .
    MESSAGE 'KMS key option is required' TYPE 'E'.
CATCH /aws1/cx_knskmstrottlingex .
    MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
CATCH /aws1/cx_knsprovthruputrexcdex .
    MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
CATCH /aws1/cx_knsresourcenotfoundex .
    MESSAGE 'Resource being access is not found.' TYPE 'E'.
ENDTRY.
```

- For API details, see [PutRecord](#) in *AWS SDK for SAP ABAP API reference*.

## Register a consumer to a Kinesis stream using an AWS SDK

The following code examples show how to register a consumer to a Kinesis stream.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// This example shows how to register a consumer to an Amazon Kinesis
/// stream. The example was written using the AWS SDK for .NET version 3.7
/// and .NET Core 5.0.
/// </summary>
public class RegisterConsumer
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();
        string consumerName = "NEW_CONSUMER_NAME";
        string streamARN = "arn:aws:kinesis:us-east-2:000000000000:stream/
AmazonKinesisStream";

        var consumer = await RegisterConsumerAsync(client, consumerName,
streamARN);

        if (consumer is not null)
        {
            Console.WriteLine($"{consumer.ConsumerName}");
        }
    }

    /// <summary>
    /// Registers the consumer to a Kinesis stream.
    /// </summary>
    /// <param name="client">The initialized Kinesis client object.</param>
    /// <param name="consumerName">A string representing the consumer.</param>
    /// <param name="streamARN">The ARN of the stream.</param>
    /// <returns>A Consumer object that contains information about the
consumer.</returns>
    public static async Task<Consumer> RegisterConsumerAsync(IAmazonKinesis
client, string consumerName, string streamARN)
    {
        var request = new RegisterStreamConsumerRequest
        {
            ConsumerName = consumerName,
            StreamARN = streamARN,
        };

        var response = await client.RegisterStreamConsumerAsync(request);
```

```
        return response.Consumer;
    }
}
```

- For API details, see [RegisterStreamConsumer](#) in *AWS SDK for .NET API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_kns->registerstreamconsumer(          " oo_result is returned
for testing purpose "
        iv_streamarn = iv_stream_arn
        iv_consumername = iv_consumer_name
    ).
    MESSAGE 'Stream consumer registered' TYPE 'I'.
    CATCH /aws1/cx_knsinvalidargumentex .
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    CATCH /aws1/cx_sgmresourceclimitexcd.
        MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
    CATCH /aws1/cx_sgmresourceinuse.
        MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
    CATCH /aws1/cx_sgmresourcenotfound.
        MESSAGE 'Resource being access is not found.' TYPE 'E'.
ENDTRY.
```

- For API details, see [RegisterStreamConsumer](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios for Kinesis using AWS SDKs

The following code examples show how to use Amazon Kinesis with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Get started with basic Kinesis data stream operations using an AWS SDK \(p. 1194\)](#)

## Get started with basic Kinesis data stream operations using an AWS SDK

The following code example shows how to:

- Create a stream.
- Put record in a stream.
- Create a shard iterator.

- Read the record.
- Delete stream.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_stream_describe_result TYPE REF TO /aws1/cl_knsdescrstreamoutput.
DATA lo_stream_description TYPE REF TO /aws1/cl_knsstreamdescription.
DATA lo_sharditerator TYPE REF TO /aws1/cl_knsgesharditerator01.
DATA lo_record_result TYPE REF TO /aws1/cl_knspputrecordoutput.

"Create stream"
TRY.
    lo_kns->createstream(
        iv_streamname = iv_stream_name
        iv_shardcount = iv_shard_count
    ).
    MESSAGE 'Stream created' TYPE 'I'.
    CATCH /aws1/cx_knsinvalidargumentex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    CATCH /aws1/cx_knslimitexceededex .
        MESSAGE 'The request processing has failed because of a limit exceed exception.' TYPE 'E'.
    CATCH /aws1/cx_knsresourceinuseex .
        MESSAGE 'The request processing has failed because the resource is in use.' TYPE 'E'.
ENDTRY.

"Wait for steam to becomes active"
lo_stream_describe_result = lo_kns->describestream( iv_streamname =
iv_stream_name ).
lo_stream_description = lo_stream_describe_result->get_streamdescription( ).
WHILE lo_stream_description->get_streamstatus( ) <> 'ACTIVE'.
    IF sy-index = 30.
        EXIT.                      "maximum 5 minutes"
    ENDIF.
    WAIT UP TO 10 SECONDS.
    lo_stream_describe_result = lo_kns->describestream( iv_streamname =
iv_stream_name ).
    lo_stream_description = lo_stream_describe_result->get_streamdescription( ).
ENDWHILE.

"Create record"
TRY.
    lo_record_result = lo_kns->putrecord(
        iv_streamname = iv_stream_name
        iv_data       = iv_data
        iv_partitionkey = iv_partition_key
    ).
    MESSAGE 'Record created' TYPE 'I'.
    CATCH /aws1/cx_knsinvalidargumentex .
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    CATCH /aws1/cx_knskmsaccessdeniedex .
```

```
MESSAGE 'You do not have permission to perform this KMS action.' TYPE 'E'.
CATCH /aws1/cx_knskmsdisabledex .
MESSAGE 'KMS key used is disabled' TYPE 'E'.
CATCH /aws1/cx_knskmsinvalidstateex .
MESSAGE 'KMS key used is in an invalid state ' TYPE 'E'.
CATCH /aws1/cx_knskmsnotfoundex .
MESSAGE 'KMS key used is not found' TYPE 'E'.
CATCH /aws1/cx_knskmsoptinrequired .
MESSAGE 'KMS key option is required' TYPE 'E'.
CATCH /aws1/cx_knskmstrottlingex .
MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
CATCH /aws1/cx_knsprovthruputexdex .
MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
CATCH /aws1/cx_knsresourcenotfoundex .
MESSAGE 'Resource being access is not found.' TYPE 'E'.
ENDTRY.

"Create a shard iterator in order to read the record"
TRY.
    lo_sharditerator = lo_kns->getsharditerator(
        iv_shardid = lo_record_result->get_shardid( )
        iv_sharditeratortype = iv_sharditeratortype
        iv_streamname = iv_stream_name
    ).
MESSAGE 'Shard iterator created' TYPE 'I'.
CATCH /aws1/cx_knsinvalidargumentex .
MESSAGE 'The specified argument was not valid.' TYPE 'E'.
CATCH /aws1/cx_knsprovthruputexdex .
MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcenotfound .
MESSAGE 'Resource being access is not found.' TYPE 'E'.
ENDTRY.

"Read the record"
TRY.
    oo_result = lo_kns->getrecords(                               " oo_result is returned
for testing purpose "
    iv_sharditerator = lo_sharditerator->get_sharditerator( )
).
MESSAGE 'Shard iterator created' TYPE 'I'.
CATCH /aws1/cx_knsexpirediteratorex .
MESSAGE 'Iterator expired.' TYPE 'E'.
CATCH /aws1/cx_knsinvalidargumentex .
MESSAGE 'The specified argument was not valid.' TYPE 'E'.
CATCH /aws1/cx_knskmsaccessdeniedex .
MESSAGE 'You do not have permission to perform this KMS action.' TYPE 'E'.
CATCH /aws1/cx_knskmsdisabledex .
MESSAGE 'KMS key used is disabled' TYPE 'E'.
CATCH /aws1/cx_knskmsinvalidstateex .
MESSAGE 'KMS key used is in an invalid state ' TYPE 'E'.
CATCH /aws1/cx_knskmsnotfoundex .
MESSAGE 'KMS key used is not found' TYPE 'E'.
CATCH /aws1/cx_knskmsoptinrequired .
MESSAGE 'KMS key option is required' TYPE 'E'.
CATCH /aws1/cx_knskmstrottlingex .
MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.'
TYPE 'E'.
CATCH /aws1/cx_knsprovthruputexdex .
MESSAGE 'The request rate for the stream is too high, or the requested data
is too large for the available throughput.' TYPE 'E'.
CATCH /aws1/cx_knsresourcenotfoundex .
MESSAGE 'Resource being access is not found.' TYPE 'E'.
ENDTRY.
```

```
"Delete Stream"
TRY.
    lo_kns->deleteteststream(
        iv_streamname = iv_stream_name
    ).
    MESSAGE 'Stream deleted' TYPE 'I'.
    CATCH /aws1/cx_knslimitexceededex .
        MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
        CATCH /aws1/cx_knsresourceinuseex .
            MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [CreateStream](#)
  - [DeleteStream](#)
  - [GetRecords](#)
  - [GetShardIterator](#)
  - [PutRecord](#)

## Code examples for Kinesis Data Analytics using AWS SDKs

The following code examples show how to use Amazon Kinesis Data Analytics with an AWS software development kit (SDK).

### Code examples

- [Actions for Kinesis Data Analytics using AWS SDKs \(p. 1198\)](#)
  - [Add an input stream to a Kinesis Data Analytics application using an AWS SDK \(p. 1198\)](#)
  - [Add an output stream to a Kinesis Data Analytics application using an AWS SDK \(p. 1199\)](#)
  - [Create a Kinesis Data Analytics application using an AWS SDK \(p. 1200\)](#)
  - [Delete a Kinesis Data Analytics application using an AWS SDK \(p. 1201\)](#)
  - [Describe a Kinesis Data Analytics application using an AWS SDK \(p. 1201\)](#)
  - [Describe a Kinesis Data Analytics application snapshot using an AWS SDK \(p. 1202\)](#)
  - [Discover a data format for a Kinesis Data Analytics stream using an AWS SDK \(p. 1203\)](#)
  - [Start a Kinesis Data Analytics application using an AWS SDK \(p. 1204\)](#)
  - [Stop a Kinesis Data Analytics application using an AWS SDK \(p. 1205\)](#)
  - [Update a Kinesis Data Analytics application using an AWS SDK \(p. 1205\)](#)
- [Data generator for Kinesis Data Analytics using AWS SDKs \(p. 1206\)](#)
  - [Generate a Kinesis stream with a referrer using an AWS SDK \(p. 1207\)](#)
  - [Generate a Kinesis stream with blood pressure anomalies using an AWS SDK \(p. 1207\)](#)
  - [Generate a Kinesis stream with data in columns using an AWS SDK \(p. 1209\)](#)
  - [Generate a Kinesis stream with heart rate anomalies using an AWS SDK \(p. 1209\)](#)
  - [Generate a Kinesis stream with hotspots using an AWS SDK \(p. 1210\)](#)
  - [Generate a Kinesis stream with log entries using an AWS SDK \(p. 1211\)](#)
  - [Generate a Kinesis stream with stagger data using an AWS SDK \(p. 1212\)](#)
  - [Generate a Kinesis stream with stock ticker data using an AWS SDK \(p. 1213\)](#)

- Generate a Kinesis stream with two data types using an AWS SDK (p. 1214)
- Generate a Kinesis stream with web log data using an AWS SDK (p. 1215)

## Actions for Kinesis Data Analytics using AWS SDKs

The following code examples show how to use Amazon Kinesis Data Analytics with AWS SDKs. Each example calls an individual service function.

### Examples

- Add an input stream to a Kinesis Data Analytics application using an AWS SDK (p. 1198)
- Add an output stream to a Kinesis Data Analytics application using an AWS SDK (p. 1199)
- Create a Kinesis Data Analytics application using an AWS SDK (p. 1200)
- Delete a Kinesis Data Analytics application using an AWS SDK (p. 1201)
- Describe a Kinesis Data Analytics application using an AWS SDK (p. 1201)
- Describe a Kinesis Data Analytics application snapshot using an AWS SDK (p. 1202)
- Discover a data format for a Kinesis Data Analytics stream using an AWS SDK (p. 1203)
- Start a Kinesis Data Analytics application using an AWS SDK (p. 1204)
- Stop a Kinesis Data Analytics application using an AWS SDK (p. 1205)
- Update a Kinesis Data Analytics application using an AWS SDK (p. 1205)

## Add an input stream to a Kinesis Data Analytics application using an AWS SDK

The following code example shows how to add an input stream to a Kinesis Data Analytics application.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:  
    """Encapsulates Kinesis Data Analytics application functions."""  
    def __init__(self, analytics_client):  
        """  
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.  
        """  
        self.analytics_client = analytics_client  
        self.name = None  
        self.arn = None  
        self.version_id = None  
        self.create_timestamp = None  
  
    def add_input(self, input_prefix, stream_arn, input_schema):  
        """  
        Adds an input stream to the application. The input stream data is mapped  
        to an in-application stream that can be processed by your code running in  
        Kinesis Data Analytics.  
  
        :param input_prefix: The prefix prepended to in-application input stream  
        names.  
        """
```

```

:param stream_arn: The ARN of the input stream.
:param input_schema: A schema that maps the data in the input stream to the
                     runtime environment. This can be automatically
                     generated
                     by using `discover_input_schema` or you can create it
                     yourself.
:return: Metadata about the newly added input.
"""
try:
    response = self.analytics_client.add_application_input(
        ApplicationName=self.name,
        CurrentApplicationVersionId=self.version_id,
        Input={
            'NamePrefix': input_prefix,
            'KinesisStreamsInput': {'ResourceARN': stream_arn},
            'InputSchema': input_schema})
    self.version_id = response['ApplicationVersionId']
    logger.info(
        "Add input stream %s to application %s.", stream_arn, self.name)
except ClientError:
    logger.exception(
        "Couldn't add input stream %s to application %s.", stream_arn,
        self.name)
    raise
else:
    return response

```

- For API details, see [AddApplicationInput](#) in *AWS SDK for Python (Boto3) API Reference*.

## Add an output stream to a Kinesis Data Analytics application using an AWS SDK

The following code example shows how to add an output stream to a Kinesis Data Analytics application.

**Python**

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""
    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None

    def add_output(self, in_app_stream_name, output_arn):
        """
        Adds an output stream to the application. Kinesis Data Analytics maps data
        from the specified in-application stream to the output stream.

        :param in_app_stream_name: The name of the in-application stream to map
        """

```

```
        to the output stream.  
:param output_arn: The ARN of the output stream.  
:return: A list of metadata about the output resources currently assigned  
        to the application.  
....  
try:  
    response = self.analytics_client.add_application_output(  
        ApplicationName=self.name,  
        CurrentApplicationVersionId=self.version_id,  
        Output=[  
            'Name': in_app_stream_name,  
            'KinesisStreamsOutput': {'ResourceARN': output_arn},  
            'DestinationSchema': {'RecordFormatType': 'JSON'}})  
    outputs = response['OutputDescriptions']  
    self.version_id = response['ApplicationVersionId']  
    logging.info(  
        "Added output %s to %s, which now has %s outputs.", output_arn,  
        self.name, len(outputs))  
except ClientError:  
    logger.exception("Couldn't add output %s to %s.", output_arn,  
self.name)  
    raise  
else:  
    return outputs
```

- For API details, see [AddApplicationOutput](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a Kinesis Data Analytics application using an AWS SDK

The following code example shows how to create a Kinesis Data Analytics application.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:  
    """Encapsulates Kinesis Data Analytics application functions."""  
    def __init__(self, analytics_client):  
        """  
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.  
        """  
        self.analytics_client = analytics_client  
        self.name = None  
        self.arn = None  
        self.version_id = None  
        self.create_timestamp = None  
  
    def create(self, app_name, role_arn, env='SQL-1_0'):  
        """  
        Creates a Kinesis Data Analytics application.  
  
        :param app_name: The name of the application.  
        :param role_arn: The ARN of a role that can be assumed by Kinesis Data  
                        Analytics and grants needed permissions.  
        :param env: The runtime environment of the application, such as SQL. Code  
                   uploaded to the application runs in this environment.  
        :return: Metadata about the newly created application.
```

```
"""
try:
    response = self.analytics_client.create_application(
        ApplicationName=app_name, RuntimeEnvironment=env,
        ServiceExecutionRole=role_arn)
    details = response['ApplicationDetail']
    self._update_details(details)
    logger.info("Application %s created.", app_name)
except ClientError:
    logger.exception("Couldn't create application %s.", app_name)
    raise
else:
    return details
```

- For API details, see [CreateApplication](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a Kinesis Data Analytics application using an AWS SDK

The following code example shows how to delete a Kinesis Data Analytics application.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""
    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None

    def delete(self):
        """
        Deletes an application.
        """
        try:
            self.analytics_client.delete_application(
                ApplicationName=self.name, CreateTimestamp=self.create_timestamp)
            logger.info("Deleted application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't delete application %s.", self.name)
            raise
```

- For API details, see [DeleteApplication](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a Kinesis Data Analytics application using an AWS SDK

The following code example shows how to describe a Kinesis Data Analytics application.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:  
    """Encapsulates Kinesis Data Analytics application functions."""  
    def __init__(self, analytics_client):  
        """  
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.  
        """  
        self.analytics_client = analytics_client  
        self.name = None  
        self.arn = None  
        self.version_id = None  
        self.create_timestamp = None  
  
    def describe(self, name):  
        """  
        Gets metadata about an application.  
  
        :param name: The name of the application to look up.  
        :return: Metadata about the application.  
        """  
        try:  
            response = self.analytics_client.describe_application(  
                ApplicationName=name)  
            details = response['ApplicationDetail']  
            self._update_details(details)  
            logger.info("Got metadata for application %s.", name)  
        except ClientError:  
            logger.exception("Couldn't get metadata for application %s.", name)  
            raise  
        else:  
            return details
```

- For API details, see [DescribeApplication](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a Kinesis Data Analytics application snapshot using an AWS SDK

The following code example shows how to describe a Kinesis Data Analytics application snapshot.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:  
    """Encapsulates Kinesis Data Analytics application functions."""  
    def __init__(self, analytics_client):
```

```
"""
:param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
"""
self.analytics_client = analytics_client
self.name = None
self.arn = None
self.version_id = None
self.create_timestamp = None

def describe_snapshot(self, application_name, snapshot_name):
    """
    Gets metadata about a previously saved application snapshot.

    :param application_name: The name of the application.
    :param snapshot_name: The name of the snapshot.
    :return: Metadata about the snapshot.
    """
    try:
        response = self.analytics_client.describe_application_snapshot(
            ApplicationName=application_name, SnapshotName=snapshot_name)
        snapshot = response['SnapshotDetails']
        logger.info(
            "Got metadata for snapshot %s of application %s.", snapshot_name,
            application_name)
    except ClientError:
        logger.exception(
            "Couldn't get metadata for snapshot %s of application %s.",
            snapshot_name, application_name)
        raise
    else:
        return snapshot
```

- For API details, see [DescribeApplicationSnapshot](#) in *AWS SDK for Python (Boto3) API Reference*.

## Discover a data format for a Kinesis Data Analytics stream using an AWS SDK

The following code example shows how to discover a data format for a Kinesis Data Analytics stream.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""
    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None

    def discover_input_schema(self, stream_arn, role_arn):
```

```
"""
Discovers a schema that maps data in a stream to a format that is usable by
an application's runtime environment. The stream must be active and have
enough data moving through it for the service to sample. The returned
schema
can be used when you add the stream as an input to the application or you
can
write your own schema.

:param stream_arn: The ARN of the stream to map.
:param role_arn: A role that lets Kinesis Data Analytics read from the
stream.
:return: The discovered schema of the data in the input stream.
"""
try:
    response = self.analytics_client.discover_input_schema(
        ResourceARN=stream_arn,
        ServiceExecutionRole=role_arn,
        InputStartingPositionConfiguration={'InputStartingPosition':
'NOW'})
    schema = response['InputSchema']
    logger.info("Discovered input schema for stream %s.", stream_arn)
except ClientError:
    logger.exception(
        "Couldn't discover input schema for stream %s.", stream_arn)
    raise
else:
    return schema
```

- For API details, see [DiscoverInputSchema in AWS SDK for Python \(Boto3\) API Reference](#).

## Start a Kinesis Data Analytics application using an AWS SDK

The following code example shows how to start a Kinesis Data Analytics application.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""
    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None

    def start(self, input_id):
        """
        Starts an application. After the application is running, it reads from the
        specified input stream and runs the application code on the incoming data.

        :param input_id: The ID of the input to read.
        """
```

```
"""
try:
    self.analytics_client.start_application(
        ApplicationName=self.name,
        RunConfiguration={
            'SqlRunConfigurations': [{
                'InputId': input_id,
                'InputStartingPositionConfiguration': {
                    'InputStartingPosition': 'NOW'}}]})
    logger.info("Started application %s.", self.name)
except ClientError:
    logger.exception("Couldn't start application %s.", self.name)
    raise
```

- For API details, see [StartApplication](#) in *AWS SDK for Python (Boto3) API Reference*.

## Stop a Kinesis Data Analytics application using an AWS SDK

The following code example shows how to stop a Kinesis Data Analytics application.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""
    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None

    def stop(self):
        """
        Stops an application. This stops the application from processing data but
        does not delete any resources.
        """
        try:
            self.analytics_client.stop_application(ApplicationName=self.name)
            logger.info("Stopping application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't stop application %s.", self.name)
            raise
```

- For API details, see [StopApplication](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update a Kinesis Data Analytics application using an AWS SDK

The following code example shows how to update a Kinesis Data Analytics application.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

This example updates the code that runs in an existing application.

```
class KinesisAnalyticsApplicationV2:  
    """Encapsulates Kinesis Data Analytics application functions."""  
    def __init__(self, analytics_client):  
        """  
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.  
        """  
        self.analytics_client = analytics_client  
        self.name = None  
        self.arn = None  
        self.version_id = None  
        self.create_timestamp = None  
  
    def update_code(self, code):  
        """  
        Updates the code that runs in the application. The code must run in the  
        runtime environment of the application, such as SQL. Application code  
        typically reads data from in-application streams and transforms it in some  
        way.  
  
        :param code: The code to upload. This completely replaces any existing code  
                    in the application.  
        :return: Metadata about the application.  
        """  
        try:  
            response = self.analytics_client.update_application(  
                ApplicationName=self.name,  
                CurrentApplicationVersionId=self.version_id,  
                ApplicationConfigurationUpdate={  
                    'ApplicationCodeConfigurationUpdate': {  
                        'CodeContentTypeUpdate': 'PLAINTEXT',  
                        'CodeContentUpdate': {  
                            'TextContentUpdate': code}}})  
            details = response['ApplicationDetail']  
            self.version_id = details['ApplicationVersionId']  
            logger.info("Update code for application %s.", self.name)  
        except ClientError:  
            logger.exception("Couldn't update code for application %s.", self.name)  
            raise  
        else:  
            return details
```

- For API details, see [UpdateApplication](#) in *AWS SDK for Python (Boto3) API Reference*.

## Data generator for Kinesis Data Analytics using AWS SDKs

The following code examples show how to use Amazon Kinesis Data Analytics with AWS SDKs.

### Examples

- [Generate a Kinesis stream with a referrer using an AWS SDK \(p. 1207\)](#)

- Generate a Kinesis stream with blood pressure anomalies using an AWS SDK (p. 1207)
- Generate a Kinesis stream with data in columns using an AWS SDK (p. 1209)
- Generate a Kinesis stream with heart rate anomalies using an AWS SDK (p. 1209)
- Generate a Kinesis stream with hotspots using an AWS SDK (p. 1210)
- Generate a Kinesis stream with log entries using an AWS SDK (p. 1211)
- Generate a Kinesis stream with stagger data using an AWS SDK (p. 1212)
- Generate a Kinesis stream with stock ticker data using an AWS SDK (p. 1213)
- Generate a Kinesis stream with two data types using an AWS SDK (p. 1214)
- Generate a Kinesis stream with web log data using an AWS SDK (p. 1215)

## Generate a Kinesis stream with a referrer using an AWS SDK

The following code example shows how to generate a Kinesis stream with a referrer.

For more information, see [Example: Extracting a portion of a stream](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import json
import boto3

STREAM_NAME = 'ExampleInputStream'

def get_data():
    return {'REFERRER': 'http://www.amazon.com'}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey='partitionkey')

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

## Generate a Kinesis stream with blood pressure anomalies using an AWS SDK

The following code example shows how to generate a Kinesis stream with blood pressure anomalies.

For more information, see [Example: Detecting data anomalies and getting an explanation](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
from enum import Enum
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

class PressureType(Enum):
    low = 'LOW'
    normal = 'NORMAL'
    high = 'HIGH'

def get_blood_pressure(pressure_type):
    pressure = {'BloodPressureLevel': pressure_type.value}
    if pressure_type == PressureType.low:
        pressure['Systolic'] = random.randint(50, 80)
        pressure['Diastolic'] = random.randint(30, 50)
    elif pressure_type == PressureType.normal:
        pressure['Systolic'] = random.randint(90, 120)
        pressure['Diastolic'] = random.randint(60, 80)
    elif pressure_type == PressureType.high:
        pressure['Systolic'] = random.randint(130, 200)
        pressure['Diastolic'] = random.randint(90, 150)
    else:
        raise TypeError
    return pressure

def generate(stream_name, kinesis_client):
    while True:
        rnd = random.random()
        pressure_type = (
            PressureType.low if rnd < 0.005
            else PressureType.high if rnd > 0.995
            else PressureType.normal)
        blood_pressure = get_blood_pressure(pressure_type)
        print(blood_pressure)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(blood_pressure),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

## Generate a Kinesis stream with data in columns using an AWS SDK

The following code example shows how to generate a Kinesis stream with data in columns.

For more information, see [Example: Split strings into multiple fields](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'Col_A': 'a',
        'Col_B': 'b',
        'Col_C': 'c',
        'Col_E_Unstructured': 'x,y,z'}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

## Generate a Kinesis stream with heart rate anomalies using an AWS SDK

The following code example shows how to generate a Kinesis stream with heart rate anomalies.

For more information, see [Example: Detecting data anomalies on a stream](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
from enum import Enum
import json
import random
import boto3

STREAM_NAME = 'ExampleInputStream'

class RateType(Enum):
    normal = 'NORMAL'
    high = 'HIGH'

def get_heart_rate(rate_type):
    if rate_type == RateType.normal:
        rate = random.randint(60, 100)
    elif rate_type == RateType.high:
        rate = random.randint(150, 200)
    else:
        raise TypeError
    return {'heartRate': rate, 'rateType': rate_type.value}

def generate(stream_name, kinesis_client, output=True):
    while True:
        rnd = random.random()
        rate_type = RateType.high if rnd < 0.01 else RateType.normal
        heart_rate = get_heart_rate(rate_type)
        if output:
            print(heart_rate)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(heart_rate),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

## Generate a Kinesis stream with hotspots using an AWS SDK

The following code example shows how to generate a Kinesis stream with hotspots.

For more information, see [Example: Detecting hotspot on a stream](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import json
from pprint import pprint
import random
```

```
import time
import boto3

STREAM_NAME = "ExampleInputStream"

def get_hotspot(field, spot_size):
    hotspot = {
        'left': field['left'] + random.random() * (field['width'] - spot_size),
        'width': spot_size,
        'top': field['top'] + random.random() * (field['height'] - spot_size),
        'height': spot_size
    }
    return hotspot

def get_record(field, hotspot, hotspot_weight):
    rectangle = hotspot if random.random() < hotspot_weight else field
    point = {
        'x': rectangle['left'] + random.random() * rectangle['width'],
        'y': rectangle['top'] + random.random() * rectangle['height'],
        'is_hot': 'Y' if rectangle is hotspot else 'N'
    }
    return {'Data': json.dumps(point), 'PartitionKey': 'partition_key'}

def generate(
        stream_name, field, hotspot_size, hotspot_weight, batch_size,
        kinesis_client):
    """
    Generates points used as input to a hotspot detection algorithm.
    With probability hotspot_weight (20%), a point is drawn from the hotspot;
    otherwise, it is drawn from the base field. The location of the hotspot
    changes for every 1000 points generated.
    """
    points_generated = 0
    hotspot = None
    while True:
        if points_generated % 1000 == 0:
            hotspot = get_hotspot(field, hotspot_size)
        records = [
            get_record(field, hotspot, hotspot_weight) for _ in range(batch_size)]
        points_generated += len(records)
        pprint(records)
        kinesis_client.put_records(StreamName=stream_name, Records=records)

        time.sleep(0.1)

if __name__ == "__main__":
    generate(
        stream_name=STREAM_NAME,
        field={'left': 0, 'width': 10, 'top': 0, 'height': 10},
        hotspot_size=1, hotspot_weight=0.2, batch_size=10,
        kinesis_client=boto3.client('kinesis'))
```

## Generate a Kinesis stream with log entries using an AWS SDK

The following code example shows how to generate a Kinesis stream with log entries.

For more information, see [Example: Parsing log strings based on regular expressions](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import json
import boto3

STREAM_NAME = 'ExampleInputStream'

def get_data():
    return [
        'LOGENTRY': '203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] '
                    '"GET /index.php HTTP/1.1" 200 125 "-" '
                    '"Mozilla/5.0 [en] Gecko/20100101 Firefox/52.0"',
    ]

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

## Generate a Kinesis stream with stagger data using an AWS SDK

The following code example shows how to generate a Kinesis stream with stagger data.

For more information, see [Example: Stagger window](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import datetime
import json
import random
import time
import boto3

STREAM_NAME = "ExampleInputStream"
```

```
def get_data():
    event_time = datetime.datetime.utcnow() - datetime.timedelta(seconds=10)
    return {
        'EVENT_TIME': event_time.isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV'])}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        # Send six records, ten seconds apart, with the same event time and ticker
        for _ in range(6):
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey")
            time.sleep(10)

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

## Generate a Kinesis stream with stock ticker data using an AWS SDK

The following code example shows how to generate a Kinesis stream with stock ticker data.

For more information, see [Examples: Windows and aggregation](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'EVENT_TIME': datetime.datetime.now().isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'PRICE': round(random.random() * 100, 2)}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
```

```
        Data=json.dumps(data),
        PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

## Generate a Kinesis stream with two data types using an AWS SDK

The following code example shows how to generate a Kinesis stream with two data types.

For more information, see [Example: Transforming multiple data types](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import json
import random
import boto3

STREAM_NAME = "OrdersAndTradesStream"
PARTITION_KEY = "partition_key"

def get_order(order_id, ticker):
    return {
        'RecordType': 'Order',
        'Oid': order_id,
        'Oticker': ticker,
        'Oprice': random.randint(500, 10000),
        'Otype': 'Sell'}

def get_trade(order_id, trade_id, ticker):
    return {
        'RecordType': "Trade",
        'Tid': trade_id,
        'Toid': order_id,
        'Tticker': ticker,
        'Tprice': random.randint(0, 3000)}

def generate(stream_name, kinesis_client):
    order_id = 1
    while True:
        ticker = random.choice(['AAAA', 'BBBB', 'CCCC'])
        order = get_order(order_id, ticker)
        print(order)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(order),
            PartitionKey=PARTITION_KEY)
        for trade_id in range(1, random.randint(0, 6)):
            trade = get_trade(order_id, trade_id, ticker)
```

```
        print(trade)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(trade),
            PartitionKey=PARTITION_KEY)
        order_id += 1

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

## Generate a Kinesis stream with web log data using an AWS SDK

The following code example shows how to generate a Kinesis stream with web log data.

For more information, see [Example: Parsing web logs](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {'log': '192.168.254.30 - John [24/May/2004:22:01:02 -0700] '
                  '"GET /icons/apache_pb.gif HTTP/1.1" 304 0'}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

## Code examples for Lambda using AWS SDKs

The following code examples show how to use AWS Lambda with an AWS software development kit (SDK).

[Code examples](#)

- [Actions for Lambda using AWS SDKs \(p. 1216\)](#)
  - [Create a Lambda function using an AWS SDK \(p. 1216\)](#)
  - [Delete a Lambda function using an AWS SDK \(p. 1222\)](#)
  - [Get a Lambda function using an AWS SDK \(p. 1226\)](#)
  - [Invoke a Lambda function using an AWS SDK \(p. 1229\)](#)
  - [List Lambda functions using an AWS SDK \(p. 1234\)](#)
  - [Update Lambda function code using an AWS SDK \(p. 1239\)](#)
  - [Update Lambda function configuration using an AWS SDK \(p. 1243\)](#)
- [Scenarios for Lambda using AWS SDKs \(p. 1246\)](#)
  - [Get started creating and invoking Lambda functions using an AWS SDK \(p. 1246\)](#)
- [Cross-service examples for Lambda using AWS SDKs \(p. 1282\)](#)
  - [Create an API Gateway REST API to track COVID-19 data \(p. 1283\)](#)
  - [Create a lending library REST API \(p. 1283\)](#)
  - [Create a messenger application with Step Functions \(p. 1284\)](#)
  - [Create a websocket chat application with API Gateway \(p. 1284\)](#)
  - [Invoke a Lambda function from a browser \(p. 1285\)](#)
  - [Use API Gateway to invoke a Lambda function \(p. 1285\)](#)
  - [Use Step Functions to invoke Lambda functions \(p. 1287\)](#)
  - [Use scheduled events to invoke a Lambda function \(p. 1288\)](#)

## Actions for Lambda using AWS SDKs

The following code examples show how to use AWS Lambda with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create a Lambda function using an AWS SDK \(p. 1216\)](#)
- [Delete a Lambda function using an AWS SDK \(p. 1222\)](#)
- [Get a Lambda function using an AWS SDK \(p. 1226\)](#)
- [Invoke a Lambda function using an AWS SDK \(p. 1229\)](#)
- [List Lambda functions using an AWS SDK \(p. 1234\)](#)
- [Update Lambda function code using an AWS SDK \(p. 1239\)](#)
- [Update Lambda function configuration using an AWS SDK \(p. 1243\)](#)

## Create a Lambda function using an AWS SDK

The following code examples show how to create a Lambda function.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Creates a new Lambda function.
///</summary>
///<param name="client">The initialized Lambda client object.</param>
///<param name="functionName">The name of the function.</param>
///<param name="s3Bucket">The S3 bucket where the zip file containing
///the code is located.</param>
///<param name="s3Key">The Amazon S3 key of the zip file.</param>
///<param name="role">A role with the appropriate Lambda
///permissions.</param>
///<param name="handler">The name of the handler function.</param>
///<returns>The Amazon Resource Name (ARN) of the newly created
///Lambda function.</returns>
public async Task<string> CreateLambdaFunction(
    AmazonLambdaClient client,
    string functionName,
    string s3Bucket,
    string s3Key,
    string role,
    string handler)
{
    var functionCode = new FunctionCode
    {
        S3Bucket = s3Bucket,
        S3Key = s3Key,
    };

    var createFunctionRequest = new CreateFunctionRequest
    {
        FunctionName = functionName,
        Description = "Created by the Lambda .NET API",
        Code = functionCode,
        Handler = handler,
        Runtime = Runtime.Dotnet6,
        Role = role,
    };

    var response = await client.CreateFunctionAsync(createFunctionRequest);
    return response.FunctionArn;
}
```

- For API details, see [CreateFunction](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createLambdaFunction(LambdaClient awsLambda,
                                         String functionName,
                                         String filePath,
                                         String role,
                                         String handler) {

    try {
        LambdaWaiter waiter = awsLambda.waiter();
        InputStream is = new FileInputStream(filePath);
        SdkBytes fileToUpload = SdkBytes.fromInputStream(is);
```

```
FunctionCode code = FunctionCode.builder()
    .zipFile(fileToUpload)
    .build();

CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
    .functionName(functionName)
    .description("Created by the Lambda Java API")
    .code(code)
    .handler(handler)
    .runtime(Runtime.JAVA8)
    .role(role)
    .build();

// Create a Lambda function using a waiter.
CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
    .functionName(functionName)
    .build();
WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("The function ARN is " +
functionResponse.functionArn());

} catch(LambdaException | FileNotFoundException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [CreateFunction](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const createFunction = async (funcName, roleArn) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const code = await zip(`.${dirname}./functions/${funcName}`);

  const command = new CreateFunctionCommand({
    Code: { ZipFile: code },
    FunctionName: funcName,
    Role: roleArn,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

- For API details, see [CreateFunction](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createNewFunction(  
    myFunctionName: String,  
    s3BucketName: String,  
    myS3Key: String,  
    myHandler: String,  
    myRole: String  
) : String? {  
  
    val functionCode = FunctionCode {  
        s3Bucket = s3BucketName  
        s3Key = myS3Key  
    }  
  
    val request = CreateFunctionRequest {  
        functionName = myFunctionName  
        code = functionCode  
        description = "Created by the Lambda Kotlin API"  
        handler = myHandler  
        role = myRole  
        runtime = Runtime.Java8  
    }  
  
    LambdaClient { region = "us-west-2" }.use { awsLambda ->  
        val functionResponse = awsLambda.createFunction(request)  
        awsLambda.waitUntilFunctionActive {  
            functionName = myFunctionName  
        }  
        return functionResponse.functionArn  
    }  
}
```

- For API details, see [CreateFunction](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function createFunction($functionName, $role, $bucketName, $handler)  
{  
    //This assumes the Lambda function is in an S3 bucket.  
    return $this->customWaiter(function () use ($functionName, $role,  
    $bucketName, $handler) {  
        return $this->lambdaClient->createFunction([  
            'Code' => [  
                'S3Bucket' => $bucketName,
```

```
        'S3Key' => $functionName,
    ],
    'FunctionName' => $functionName,
    'Role' => $role['Arn'],
    'Runtime' => 'python3.9',
    'Handler' => "$handler.lambda_handler",
]);
}
}
```

- For API details, see [CreateFunction](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    def create_function(self, function_name, handler_name, iam_role,
                       deployment_package):
        """
        Deploys a Lambda function.

        :param function_name: The name of the Lambda function.
        :param handler_name: The fully qualified name of the handler function. This
                             must include the file name and the function name.
        :param iam_role: The IAM role to use for the function.
        :param deployment_package: The deployment package that contains the
                                  function
                                         code in .zip format.
        :return: The Amazon Resource Name (ARN) of the newly created function.
        """
        try:
            response = self.lambda_client.create_function(
                FunctionName=function_name,
                Description="AWS Lambda doc example",
                Runtime='python3.8',
                Role=iam_role.arn,
                Handler=handler_name,
                Code={'ZipFile': deployment_package},
                Publish=True)
            function_arn = response['FunctionArn']
            waiter = self.lambda_client.get_waiter('function_active_v2')
            waiter.wait(FunctionName=function_name)
            logger.info("Created function '%s' with ARN: '%s'.",
                        function_name, response['FunctionArn'])
        except ClientError:
            logger.error("Couldn't create function %s.", function_name)
            raise
        else:
            return function_arn
```

- For API details, see [CreateFunction](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function. This
  #                      must include the file name and the function name.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
  #                           code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
      role: role_arn.to_s,
      function_name:,
      handler: handler_name,
      runtime: "ruby2.7",
      code: {
        zip_file: deployment_package
      },
      environment: {
        variables: {
          "LOG_LEVEL" => "info"
        }
      }
    })
    @lambda_client.wait_until(:function_active_v2, { function_name: }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n#{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n#{e.message}")
  end
end
```

- For API details, see [CreateFunction](#) in *AWS SDK for Ruby API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_lmd->createfunction(  
        iv_functionname = iv_function_name  
        iv_runtime = `python3.9'  
        iv_role = iv_role_arn  
        iv_handler = iv_handler  
        io_code = io_zip_file  
        iv_description = 'AWS Lambda code example'  
    ).  
    MESSAGE 'Lambda function created' TYPE 'I'.  
    CATCH /aws1/cx_lmdcodesigningcfgno00.  
        MESSAGE 'Code signing configuration does not exist' TYPE 'E'.  
    CATCH /aws1/cx_lmdcodestorageexcdex.  
        MESSAGE 'Maximum total code size per account exceeded' TYPE 'E'.  
    CATCH /aws1/cx_lmdcodeverification00.  
        MESSAGE 'Code signature failed one or more validation checks for signature mismatch or expiration' TYPE 'E'.  
    CATCH /aws1/cx_lmdinvalidcodesigex.  
        MESSAGE 'Code signature failed the integrity check' TYPE 'E'.  
    CATCH /aws1/cx_lmdinvparamvalueex.  
        MESSAGE 'The request contains an invalid parameter' TYPE 'E'.  
    CATCH /aws1/cx_lmdresourceconflictex.  
        MESSAGE 'Resource already exists or another operation is in progress' TYPE 'E'.  
    CATCH /aws1/cx_lmdresourcenotfoundex.  
        MESSAGE 'The requested resource does not exist' TYPE 'E'.  
    CATCH /aws1/cx_lmdserviceexception.  
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'  
    TYPE 'E'.  
    CATCH /aws1/cx_lmdtoomanyrequestsex.  
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [CreateFunction](#) in *AWS SDK for SAP ABAP API reference*.

## Delete a Lambda function using an AWS SDK

The following code examples show how to delete a Lambda function.

.NET

### AWS SDK for .NET

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Deletes an AWS Lambda function.  
/// </summary>  
/// <param name="client">An initialized Lambda client object.</param>  
/// <param name="functionName">The name of the Lambda function to  
/// delete.</param>  
/// <returns>A Boolean value that indicates where the function was
```

```
    /// successfully deleted.</returns>
    public async Task<bool> DeleteLambdaFunction(AmazonLambdaClient client,
string functionName)
{
    var request = new DeleteFunctionRequest
    {
        FunctionName = functionName,
    };

    var response = await client.DeleteFunctionAsync(request);

    // A return value of NoContent means that the request was processed
    // (in this case, the function was deleted, and the return value
    // is intentionally blank.
    return response.HttpStatusCode == System.Net.HttpStatusCode.NoContent;
}
```

- For API details, see [DeleteFunction](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("The " +functionName +" function was deleted");

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteFunction](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const deleteFunction = (funcName) => {
    const client = createClientForDefaultRegion(LambdaClient);
    const command = new DeleteFunctionCommand({ FunctionName: funcName });
```

```
    return client.send(command);
};
```

- For API details, see [DeleteFunction](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun delLambdaFunction(myFunctionName: String) {

    val request = DeleteFunctionRequest {
        functionName = myFunctionName
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- For API details, see [DeleteFunction](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function deleteFunction($functionName)
{
    return $this->lambdaClient->deleteFunction([
        'FunctionName' => $functionName,
    ]);
}
```

- For API details, see [DeleteFunction](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    def delete_function(self, function_name):
        """
        Deletes a Lambda function.

        :param function_name: The name of the function to delete.
        """
        try:
            self.lambda_client.delete_function(FunctionName=function_name)
        except ClientError:
            logger.exception("Couldn't delete function %s.", function_name)
            raise
```

- For API details, see [DeleteFunction](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deletes a Lambda function.
  # @param function_name: The name of the function to delete.
  def delete_function(function_name)
    print "Deleting function: #{function_name}..."
    @lambda_client.delete_function(
      function_name:
    )
    print "Done!".green
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error deleting #{function_name}:\n#{e.message}")
  end
```

- For API details, see [DeleteFunction](#) in *AWS SDK for Ruby API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_lmd->deletefunction( iv_functionname = iv_function_name ).  
    MESSAGE 'Lambda function deleted' TYPE 'I'.  
CATCH /aws1/cx_lmdinvparamvalueex.  
    MESSAGE 'The request contains an invalid parameter' TYPE 'E'.  
CATCH /aws1/cx_lmdresourceconflictex.  
    MESSAGE 'Rresource already exists or another operation is in progress' TYPE  
'E'.  
CATCH /aws1/cx_lmdresourcenotfoundex.  
    MESSAGE 'The requested resource does not exist' TYPE 'E'.  
CATCH /aws1/cx_lmdserviceexception.  
    MESSAGE 'An internal problem was encountered by the AWS Lambda service.'  
TYPE 'E'.  
CATCH /aws1/cx_lmdtoomanyrequestsex.  
    MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteFunction](#) in *AWS SDK for SAP ABAP API reference*.

## Get a Lambda function using an AWS SDK

The following code examples show how to get a Lambda function.

### .NET

#### AWS SDK for .NET

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>  
/// Gets information about a Lambda function.  
///</summary>  
///<param name="client">The initialized Lambda client object.</param>  
///<param name="functionName">The name of the Lambda function for  
/// which to retrieve information.</param>  
///<returns>A System Threading Task.</returns>  
public async Task<FunctionConfiguration> GetFunction(AmazonLambdaClient  
client, string functionName)  
{  
    var functionRequest = new GetFunctionRequest  
    {  
        FunctionName = functionName,  
    };  
  
    var response = await client.GetFunctionAsync(functionRequest);  
    return response.Configuration;  
}
```

- For API details, see [GetFunction](#) in *AWS SDK for .NET API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getFunction = (funcName) => {
    const client = createClientForDefaultRegion(LambdaClient);
    const command = new GetFunctionCommand({ FunctionName: funcName });
    return client.send(command);
};
```

- For API details, see [GetFunction](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function getFunction($functionName)
{
    return $this->lambdaClient->getFunction([
        'FunctionName' => $functionName,
    ]);
}
```

- For API details, see [GetFunction](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    def get_function(self, function_name):
        """
        Gets data about a Lambda function.

        :param function_name: The name of the function.
        :return: The function data.
        """
        response = None
        try:
            response = self.lambda_client.get_function(FunctionName=function_name)
```

```
        except ClientError as err:
            if err.response['Error']['Code'] == 'ResourceNotFoundException':
                logger.info("Function %s does not exist.", function_name)
            else:
                logger.error(
                    "Couldn't get function %s. Here's why: %s: %s",
                    function_name,
                    err.response['Error']['Code'], err.response['Error']
                )
            raise
    return response
```

- For API details, see [GetFunction in AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Gets data about a Lambda function.
  #
  # @param function_name: The name of the function.
  # @return response: The function data, or nil if no such function exists.
  def get_function(function_name)
    @lambda_client.get_function(
      {
        function_name:
      }
    )
    rescue Aws::Lambda::Errors::ResourceNotFoundException => e
      @logger.debug("Could not find function: #{function_name}:\n#{e.message}")
      nil
  end
```

- For API details, see [GetFunction in AWS SDK for Ruby API Reference](#).

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_lmd->getfunction( iv_functionname = iv_function_name ).  
" oo_result is returned for testing purpose "  
    MESSAGE 'Lambda function information retrieved' TYPE 'I'.  
CATCH /aws1/cx_lmdinparamvalueex.  
    MESSAGE 'The request contains an invalid parameter' TYPE 'E'.  
CATCH /aws1/cx_lmdserviceexception.  
    MESSAGE 'An internal problem was encountered by the AWS Lambda service.'  
TYPE 'E'.  
CATCH /aws1/cx_lmdtoomanyrequestsex.  
    MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [GetFunction](#) in *AWS SDK for SAP ABAP API reference*.

## Invoke a Lambda function using an AWS SDK

The following code examples show how to invoke a Lambda function.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System.Threading.Tasks;  
using Amazon.Lambda;  
using Amazon.Lambda.Model;  
  
/// <summary>  
/// Shows how to invoke an existing Amazon Lambda Function from a C#  
/// application. The example was created using the AWS SDK for .NET and  
/// .NET Core 5.0.  
/// </summary>  
public class InvokeFunction  
{  
    /// <summary>  
    /// Initializes the Lambda client and then invokes the Lambda Function  
    /// called "CreateDDBTable" with the parameter "\"DDBWorkTable\""  
    /// to  
    /// create the table called DDBWorkTable.  
    /// </summary>  
    public static async Task Main()  
    {  
        IAmazonLambda client = new AmazonLambdaClient();  
        string functionName = "CreateDDBTable";  
        string invokeArgs = "\"DDBWorkTable\"";  
  
        var response = await client.InvokeAsync(  
            new InvokeRequest  
            {  
                FunctionName = functionName,  
                Payload = invokeArgs,  
                InvocationType = "Event",  
            });  
    }  
}
```

```
/// <summary>
/// Invokes a Lambda function.
/// </summary>
/// <param name="client">An initialized Lambda client object.</param>
/// <param name="functionName">The name of the Lambda function to
/// invoke.</param>
/// <returns>A System Threading Task.</returns>
public async Task<string> InvokeFunctionAsync(
    AmazonLambdaClient client,
    string functionName,
    string parameters)
{
    var payload = parameters;
    var request = new InvokeRequest
    {
        FunctionName = functionName,
        Payload = payload,
    };

    var response = await client.InvokeAsync(request);
    MemoryStream stream = response.Payload;
    string returnValue =
System.Text.Encoding.UTF8.GetString(stream.ToArray());
    return returnValue;
}
```

- For API details, see [Invoke](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void invokeFunction(LambdaClient awsLambda, String functionName)
{
    InvokeResponse res = null ;
    try {
        // Need a SdkBytes instance for the payload.
        JSONObject jsonObj = new JSONObject();
        jsonObj.put("inputValue", "2000");
        String json = jsonObj.toString();
        SdkBytes payload = SdkBytes.fromUtf8String(json) ;

        // Setup an InvokeRequest.
        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String() ;
        System.out.println(value);

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}
```

- For API details, see [Invoke](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const invoke = async (funcName, payload) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const command = new InvokeCommand({
    FunctionName: funcName,
    Payload: JSON.stringify(payload),
    LogType: LogType.Tail,
  });

  const { Payload, LogResult } = await client.send(command);
  const result = Buffer.from(Payload).toString();
  const logs = Buffer.from(LogResult, "base64").toString();
  return { logs, result };
};
```

- For API details, see [Invoke](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun invokeFunction(functionNameVal: String) {

    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request = InvokeRequest {
        functionName = functionNameVal
        logType = LogType.Tail
        payload = byteArray
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("${res.payload?.toString(Charsets.UTF_8)}")
        println("The log result is ${res.logResult}")
    }
}
```

- For API details, see [Invoke](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function invoke($functionName, $params, $logType = 'None')
{
    return $this->lambdaClient->invoke([
        'FunctionName' => $functionName,
        'Payload' => json_encode($params),
        'LogType' => $logType,
    ]);
}
```

- For API details, see [Invoke](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    def invoke_function(self, function_name, function_params, get_log=False):
        """
        Invokes a Lambda function.

        :param function_name: The name of the function to invoke.
        :param function_params: The parameters of the function as a dict. This dict
                               is serialized to JSON before it is sent to Lambda.
        :param get_log: When true, the last 4 KB of the execution log are included
                       in
                       the response.
        :return: The response from the function invocation.
        """
        try:
            response = self.lambda_client.invoke(
                FunctionName=function_name,
                Payload=json.dumps(function_params),
                LogType='Tail' if get_log else 'None')
            logger.info("Invoked function %s.", function_name)
        except ClientError:
            logger.exception("Couldn't invoke function %s.", function_name)
            raise
        return response
```

- For API details, see [Invoke](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n#{e.message}")
  end
```

- For API details, see [Invoke](#) in *AWS SDK for Ruby API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  DATA(lv_json) = /aws1/cl_rt_util=>string_to_xstring(
    ` ` &&
    ` "action": "increment", ` &&
    ` "number": 10` &&
    ` `
  ).
  oo_result =  lo_lmd->invoke(                               " oo_result is returned for
testing purpose "
    iv_functionname = iv_function_name
    iv_payload = lv_json
  ).
MESSAGE 'Lambda function invoked' TYPE 'I'.
```

```
CATCH /aws1/cx_lmdinvparamvalueex.  
    MESSAGE 'The request contains an invalid parameter' TYPE 'E'.  
CATCH /aws1/cx_lmdinvrequestcontext.  
    MESSAGE 'Unable to parse request body as JSON' TYPE 'E'.  
CATCH /aws1/cx_lmdinvalidzipfileex.  
    MESSAGE 'The deployment package could not be unzipped' TYPE 'E'.  
CATCH /aws1/cx_lmdrequesttoolargeex.  
    MESSAGE 'Invoke request body JSON input limit was exceeded by the request  
payload.' TYPE 'E'.  
CATCH /aws1/cx_lmdresourceconflictex.  
    MESSAGE 'Resource already exists or another operation is in progress' TYPE  
'E'.  
CATCH /aws1/cx_lmdresourcenotfoundex.  
    MESSAGE 'The requested resource does not exist' TYPE 'E'.  
CATCH /aws1/cx_lmdserviceexception.  
    MESSAGE 'An internal problem was encountered by the AWS Lambda service'  
TYPE 'E'.  
CATCH /aws1/cx_lmdtoomanyrequestsex.  
    MESSAGE 'The maximum request throughput was reached' TYPE 'E'.  
CATCH /aws1/cx_lmdunsupportedmediatype00.  
    MESSAGE 'Invoke request body does not have JSON as its content type' TYPE  
'E'.  
ENDTRY.
```

- For API details, see [Invoke](#) in *AWS SDK for SAP ABAP API reference*.

## List Lambda functions using an AWS SDK

The following code examples show how to list Lambda functions.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Collections.Generic;  
using System.Threading.Tasks;  
using Amazon;  
using Amazon.Lambda;  
using Amazon.Lambda.Model;  
  
/// <summary>  
/// This example shows two ways to list the AWS Lambda functions you have  
/// created for your account. It will only list the functions within one  
/// AWS Region at a time, however, so you need to pass the AWS Region you  
/// are interested in to the Lambda client object constructor. This example  
/// was created with the AWS SDK for .NET version 3.7 and .NET Core 5.0.  
/// </summary>  
public class ListFunctions  
{  
    public static async Task Main()  
    {  
        // If the AWS Region you are interested in listing is the same as  
        // the AWS Region defined for the default user, you don't have to  
        // supply the RegionEndpoint constant to the constructor.  
        IAmazonLambda client = new AmazonLambdaClient(RegionEndpoint.USEast2);
```

```
// First use the ListFunctionsAsync method.  
var functions1 = await ListFunctionsAsync(client);  
  
DisplayFunctionList(functions1);  
  
// Get the list again useing a Lambda client paginator.  
var functions2 = await ListFunctionsPaginatorAsync(client);  
  
DisplayFunctionList(functions2);  
}  
  
/// <summary>  
/// Calls the asynchronous ListFunctionsAsync method of the Lambda  
/// client to retrieve the list of functions in the AWS Region with  
/// which the Lambda client was initialized.  
/// </summary>  
/// <param name="client">The initialized Lambda client which will be  
/// used to retrieve the list of Lambda functions.</param>  
/// <returns>A list of Lambda functions configuration information.</returns>  
public static async Task<List<FunctionConfiguration>>  
ListFunctionsAsync(IAmazonLambda client)  
{  
    // Get the list of functions. The response will have a property  
    // called Functions, a list of information about the Lambda  
    // functions defined on your account in the specified Region.  
    var response = await client.ListFunctionsAsync();  
  
    return response.Functions;  
}  
  
/// <summary>  
/// Uses a Lambda paginator to retrieve the list of functions in the  
/// AWS Region with which the Lambda client was initialized.  
/// </summary>  
/// <param name="client">The initialized Lambda client which will be  
/// used to retrieve the list of Lambda functions.</param>  
/// <returns>A list of Lambda functions configuration information.</returns>  
public static async Task<List<FunctionConfiguration>>  
ListFunctionsPaginatorAsync(IAmazonLambda client)  
{  
    Console.WriteLine("\nNow let's show the list using a paginator.\n");  
  
    // Get the list of functions using a paginator.  
    var paginator = client.Paginator.ListFunctions(new  
ListFunctionsRequest());  
  
    // Defined return a list of function information to the caller  
    // for display using the DisplayFunctionList method.  
    var functions = new List<FunctionConfiguration>();  
  
    await foreach (var resp in paginator.Responses)  
    {  
        resp.Functions  
            .ForEach(f => functions.Add(f));  
    }  
  
    return functions;  
}  
  
/// <summary>  
/// Displays the details of each function in the list of functions  
/// passed to the method.  
/// </summary>
```

```

    ///</summary>
    ///<param name="functions">A list of FunctionConfiguration objects.</param>
public static void DisplayFunctionList(List<FunctionConfiguration> functions)
{
    // Display a list of the Lambda functions on the console.
    functions
        .ForEach(f => Console.WriteLine($"{f.FunctionName}\t{f.Handler}"));
}
}

///<summary>
/// Gets a list of Lambda functions.
///</summary>
///<param name="client">The initialized Lambda client object.</param>
///<returns>A list of FunctionConfiguration objects.</returns>
public async Task<List<FunctionConfiguration>>
ListFunctions(AmazonLambdaClient client)
{
    var reponse = await client.ListFunctionsAsync();
    var functionList = reponse.Functions;
    return functionList;
}

```

List functions using a paginator.

```

    ///<summary>
    /// Uses a Lambda paginator to retrieve the list of functions in the
    /// AWS Region with which the Lambda client was initialized.
    ///</summary>
    ///<param name="client">The initialized Lambda client which will be
    /// used to retrieve the list of Lambda functions.</param>
    ///<returns>A list of Lambda functions configuration information.</returns>
public static async Task<List<FunctionConfiguration>>
ListFunctionsPaginatorAsync(IAmazonLambda client)
{
    Console.WriteLine("\nNow let's show the list using a paginator.\n");

    // Get the list of functions using a paginator.
    var paginator = client.Paginator.ListFunctions(new
ListFunctionsRequest());

    // Defined return a list of function information to the caller
    // for display using the DisplayFunctionList method.
    var functions = new List<FunctionConfiguration>();

    await foreach (var resp in paginator.Responses)
    {
        resp.Functions
            .ForEach(f => functions.Add(f));
    }

    return functions;
}

```

- For API details, see [ListFunctions](#) in [AWS SDK for .NET API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const listFunctions = async () => {
  const client = createClientForDefaultRegion(LambdaClient);
  const command = new ListFunctionsCommand({});

  return client.send(command);
};
```

- For API details, see [ListFunctions](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function listFunctions($maxItems = 50, $marker = null)
{
    if (is_null($marker)) {
        return $this->lambdaClient->listFunctions([
            'MaxItems' => $maxItems,
        ]);
    }

    return $this->lambdaClient->listFunctions([
        'Marker' => $marker,
        'MaxItems' => $maxItems,
    ]);
}
```

- For API details, see [ListFunctions](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    def list_functions(self):
```

```
"""
Lists the Lambda functions for the current account.
"""
try:
    funcPaginator = self.lambda_client.getPaginator('list_functions')
    for funcPage in funcPaginator.paginate():
        for func in funcPage['Functions']:
            print(func['FunctionName'])
            desc = func.get('Description')
            if desc:
                print(f"\t{desc}")
            print(f"\t{func['Runtime']}: {func['Handler']}")
except ClientError as err:
    logger.error(
        "Couldn't list functions. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
```

- For API details, see [ListFunctions](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Lists the Lambda functions for the current account.
  def list_functions
    functions = []
    @lambda_client.list_functions.each do |response|
      response["functions"].each do |function|
        functions.append(function["function_name"])
      end
    end
    functions
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n#{e.message}")
  end
```

- For API details, see [ListFunctions](#) in *AWS SDK for Ruby API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_lmd->listfunctions( ).           " oo_result is returned for  
testing purpose "  
    DATA(lt_functions) = oo_result->get_functions( ).  
    MESSAGE 'Retrieved list of lambda function(s)' TYPE 'I'.  
CATCH /aws1/cx_lmdinvparamvalueex.  
    MESSAGE 'The request contains an invalid parameter' TYPE 'E'.  
CATCH /aws1/cx_lmdserviceexception.  
    MESSAGE 'An internal problem was encountered by the AWS Lambda service.'  
TYPE 'E'.  
CATCH /aws1/cx_lmdtoomanyrequestsex.  
    MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [ListFunctions](#) in *AWS SDK for SAP ABAP API reference*.

## Update Lambda function code using an AWS SDK

The following code examples show how to update Lambda function code.

### .NET

#### AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Updates an existing Lambda function.  
/// </summary>  
/// <param name="client">An initialized Lambda client object.</param>  
/// <param name="functionName">The name of the Lambda function to update.</param>  
/// <param name="bucketName">The bucket where the zip file containing  
/// the Lambda function code is stored.</param>  
/// <param name="key">The key name of the source code file.</param>  
/// <returns>A System Threading Task.</returns>  
public async Task UpdateFunctionCode(  
    AmazonLambdaClient client,  
    string functionName,  
    string bucketName,  
    string key)  
{  
    var functionCodeRequest = new UpdateFunctionCodeRequest  
    {  
        FunctionName = functionName,  
        Publish = true,  
        S3Bucket = bucketName,  
        S3Key = key,  
    };  
  
    var response = await  
    client.UpdateFunctionCodeAsync(functionCodeRequest);
```

```
        Console.WriteLine($"The Function was last modified at  
{response.LastModified}.");
    }
```

- For API details, see [UpdateFunctionCode in AWS SDK for .NET API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const updateFunctionCode = async (funcName, newFunc) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const code = await zip(`.${dirname}/functions/${newFunc}`);
  const command = new UpdateFunctionCodeCommand({
    ZipFile: code,
    FunctionName: funcName,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

- For API details, see [UpdateFunctionCode in AWS SDK for JavaScript API Reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function updateFunctionCode($functionName, $s3Bucket, $s3Key)
{
    return $this->lambdaClient->updateFunctionCode([
        'FunctionName' => $functionName,
        'S3Bucket' => $s3Bucket,
        'S3Key' => $s3Key,
    ]);
}
```

- For API details, see [UpdateFunctionCode in AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    def update_function_code(self, function_name, deployment_package):
        """
        Updates the code for a Lambda function by submitting a .zip archive that
        contains
            the code for the function.

        :param function_name: The name of the function to update.
        :param deployment_package: The function code to update, packaged as bytes
        in
            .zip format.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_code(
                FunctionName=function_name, ZipFile=deployment_package)
        except ClientError as err:
            logger.error(
                "Couldn't update function %s. Here's why: %s: %s",
                function_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

- For API details, see [UpdateFunctionCode](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the code for a Lambda function by submitting a .zip archive that
  # contains
  #     the code for the function.

  # @param function_name: The name of the function to update.
```

```
# @param deployment_package: The function code to update, packaged as bytes in
#                   .zip format.
# @return: Data about the update, including the status.
def update_function_code(function_name, deployment_package)
  @lambda_client.update_function_code(
    function_name:,
    zip_file: deployment_package
  )
  @lambda_client.wait_until(:function_updated_v2, { function_name: }) do |w|
    w.max_attempts = 5
    w.delay = 5
  end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for: #{function_name}:
\\n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\\n #{e.message}")
  end
```

- For API details, see [UpdateFunctionCode in AWS SDK for Ruby API Reference](#).

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  oo_result = lo_lmd->updatefunctioncode(      " oo_result is returned for
testing purpose "
    iv_functionname = iv_function_name
    iv_zipfile = io_zip_file
  ).

  MESSAGE 'Lambda function code updated' TYPE 'I'.
  CATCH /aws1/cx_lmdcodesigningcfgno00.
    MESSAGE 'Code signing configuration does not exist' TYPE 'E'.
  CATCH /aws1/cx_lmdcodestorageexcex.
    MESSAGE 'Maximum total code size per account exceeded' TYPE 'E'.
  CATCH /aws1/cx_lmdcodeverification00.
    MESSAGE 'Code signature failed one or more validation checks for signature
mismatch or expiration' TYPE 'E'.
  CATCH /aws1/cx_lmdinvalidcodesigex.
    MESSAGE 'Code signature failed the integrity check' TYPE 'E'.
  CATCH /aws1/cx_lmdinvparamvalueex.
    MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
  CATCH /aws1/cx_lmdresourceconflictex.
    MESSAGE 'Resource already exists or another operation is in progress' TYPE
'E'.
  CATCH /aws1/cx_lmdresourcenotfoundex.
    MESSAGE 'The requested resource does not exist' TYPE 'E'.
  CATCH /aws1/cx_lmdserviceexception.
    MESSAGE 'An internal problem was encountered by the AWS Lambda service.'
TYPE 'E'.
```

```
CATCH /aws1/cx_lmdtoomanyrequestsex.  
      MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [UpdateFunctionCode](#) in *AWS SDK for SAP ABAP API reference*.

## Update Lambda function configuration using an AWS SDK

The following code examples show how to update Lambda function configuration.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public async Task<bool> UpdateFunctionConfigurationAsync(  
    AmazonLambdaClient client,  
    string functionName,  
    string functionHandler,  
    Dictionary<string, string> environmentVariables)  
{  
    var request = new UpdateFunctionConfigurationRequest  
    {  
        Handler = functionHandler,  
        FunctionName = functionName,  
        Environment = new Amazon.Lambda.Model.Environment { Variables =  
            environmentVariables },  
    };  
  
    var response = await client.UpdateFunctionConfigurationAsync(request);  
  
    Console.WriteLine(response.LastModified);  
  
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;  
}
```

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for .NET API Reference*.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const updateFunctionConfiguration = async (funcName) => {  
    const client = createClientForDefaultRegion(LambdaClient);  
    const config = readFileSync(  
        `${dirname}../functions/${funcName}/config.json`  
    ).toString();  
    const command = new UpdateFunctionConfigurationCommand({
```

```
    ...JSON.parse(config),
    FunctionName: funcName,
});
return client.send(command);
};
```

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for JavaScript API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function updateFunctionConfiguration($functionName, $handler,
$environment = '')
{
    return $this->lambdaClient->updateFunctionConfiguration([
        'FunctionName' => $functionName,
        'Handler' => "$handler.lambda_handler",
        'Environment' => $environment,
    ]);
}
```

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    def update_function_configuration(self, function_name, env_vars):
        """
        Updates the environment variables for a Lambda function.

        :param function_name: The name of the function to update.
        :param env_vars: A dict of environment variables to update.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_configuration(
                FunctionName=function_name, Environment={'Variables': env_vars})
        except ClientError as err:
            logger.error(
                "Couldn't update function configuration %s. Here's why: %s: %s",
                function_name,
                err.response['Error']['Code'], err.response['Error']['Message'])

```

```
        raise
    else:
        return response
```

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
      function_name:,
      environment: {
        variables: {
          "LOG_LEVEL" => log_level
        }
      }
    })
    @lambda_client.wait_until(:function_updated_v2, { function_name: }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    rescue Aws::Lambda::Errors::ServiceException => e
      @logger.error("There was an error updating configurations for #{function_name}:\n#{e.message}")
    rescue Aws::Waiters::Errors::WaiterFailed => e
      @logger.error("Failed waiting for #{function_name} to activate:\n#{e.message}")
    end
  end
end
```

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for Ruby API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_lmd->updatefunctionconfiguration(      " oo_result is  
returned for testing purpose "  
        iv_functionname = iv_function_name  
        iv_runtime = iv_runtime  
        iv_description = 'Updated Lambda Function'  
        iv_memorysize = iv_memory_size  
    ).  
  
    MESSAGE 'Lambda function configuration/settings updated' TYPE 'I'.  
    CATCH /aws1/cx_lmdcodesigningcfgno00.  
        MESSAGE 'Code signing configuration does not exist' TYPE 'E'.  
    CATCH /aws1/cx_lmdcodeverification00.  
        MESSAGE 'Code signature failed one or more validation checks for signature  
mismatch or expiration' TYPE 'E'.  
    CATCH /aws1/cx_lmdinvalidcodesigex.  
        MESSAGE 'Code signature failed the integrity check' TYPE 'E'.  
    CATCH /aws1/cx_lmdinvalparamvalueex.  
        MESSAGE 'The request contains an invalid parameter' TYPE 'E'.  
    CATCH /aws1/cx_lmdresourceconflictex.  
        MESSAGE 'Rresource already exists or another operation is in progress' TYPE  
'E'.  
    CATCH /aws1/cx_lmdresourcenotfoundex.  
        MESSAGE 'The requested resource does not exist' TYPE 'E'.  
    CATCH /aws1/cx_lmdserviceexception.  
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.'  
TYPE 'E'.  
    CATCH /aws1/cx_lmdtoomanyrequestsex.  
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios for Lambda using AWS SDKs

The following code examples show how to use AWS Lambda with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Get started creating and invoking Lambda functions using an AWS SDK \(p. 1246\)](#)

## Get started creating and invoking Lambda functions using an AWS SDK

The following code examples show how to:

- Create an AWS Identity and Access Management (IAM) role that grants Lambda permission to write to logs.
- Create a Lambda function and upload handler code.
- Invoke the function with a single parameter and get results.

- Update the function code and configure its Lambda environment with an environment variable.
- Invoke the function with new parameters and get results. Display the execution log that's returned from the invocation.
- List the functions for your account.
- Delete the IAM role and the Lambda function.

For more information, see [Create a Lambda function with the console](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
global using Amazon;
global using Amazon.Lambda;
global using Amazon.Lambda.Model;
global using Amazon.IdentityManagement;
global using Amazon.IdentityManagement.Model;
global using Lambda_Basics;
global using Microsoft.Extensions.Configuration;

// The following variables will be loaded from a configuration file:
//
//   functionName - The name of the Lambda function.
//   roleName - The IAM service role that has Lambda permissions.
//   handler - The fully qualified method name (for example,
//             example.Handler::handleRequest).
//   bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name
//               that contains the .zip or .jar used to update the Lambda function's code.
//   key - The Amazon S3 key name that represents the .zip or .jar (for
//         example, LambdaHello-1.0-SNAPSHOT.jar).
//   keyUpdate - The Amazon S3 key name that represents the updated .zip (for
//              example, "updated-function.zip").

var configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

string functionName = configuration["FunctionName"];
string roleName = configuration["RoleName"];
string policyDocument = "{" +
    "\"Version\": \"2012-10-17\"," +
    "\"Statement\": [ " +
        "{" +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
                "\"Service\": \"lambda.amazonaws.com\" " +
            "}, " +
            "\"Action\": \"sts:AssumeRole\"," +
        "}" +
    "]," +
"}";

var incrementHandler = configuration["IncrementHandler"];
```

```
var calculatorHandler = configuration["CalculatorHandler"];
var bucketName = configuration["BucketName"];
var key = configuration["Key"];
var updateKey = configuration["UpdateKey"];

string sepBar = new('-', 80);

var lambdaClient = new AmazonLambdaClient();
var lambdaMethods = new LambdaMethods();
var lambdaRoleMethods = new LambdaRoleMethods();

ShowOverview();

// Create the policy to use with the Lambda functions and then attach the
// policy to a new role.
var roleArn = await lambdaRoleMethods.CreateLambdaRole(roleName, policyDocument);

Console.WriteLine("Waiting for role to become active.");
System.Threading.Thread.Sleep(10000);

// Create the Lambda function using a zip file stored in an S3 bucket.
Console.WriteLine(sepBar);
Console.WriteLine($"Creating the AWS Lambda function: {functionName}.");
var lambdaArn = await lambdaMethods.CreateLambdaFunction(
    lambdaClient,
    functionName,
    bucketName,
    key,
    roleArn,
    incrementHandler);

Console.WriteLine(sepBar);
Console.WriteLine($"The AWS Lambda ARN is {lambdaArn}");

// Get the Lambda function.
Console.WriteLine($"Getting the {functionName} AWS Lambda function.");
FunctionConfiguration config;
do
{
    config = await lambdaMethods.GetFunction(lambdaClient, functionName);
    Console.Write(".");
}
while (config.State != State.Active);

Console.WriteLine($"\\nThe function, {functionName} has been created.");
Console.WriteLine($"The runtime of this Lambda function is {config.Runtime}.");

PressEnter();

// List the Lambda functions.
Console.WriteLine(sepBar);
Console.WriteLine("Listing all Lambda functions.");
var functions = await lambdaMethods.ListFunctions(lambdaClient);
DisplayFunctionList(functions);
Console.WriteLine(sepBar);

Console.WriteLine(sepBar);
Console.WriteLine("Invoke the Lambda increment function.");
string? value;
do
{
    Console.Write("Enter a value to increment: ");
    value = Console.ReadLine();
}
while (value == string.Empty);
```

```
string functionParameters = "{" +
    "\"action\": \"increment\", " +
    "\"x\": \"" + value + "\" " +
"}";
var answer = await lambdaMethods.InvokeFunctionAsync(lambdaClient, functionName,
    functionParameters);
Console.WriteLine($"{value} + 1 = {answer}.");

Console.WriteLine(sepBar);
Console.WriteLine("Now update the Lambda function code.");
await lambdaMethods.UpdateFunctionCode(lambdaClient, functionName, bucketName,
    updateKey);

do
{
    config = await lambdaMethods.GetFunction(lambdaClient, functionName);
    Console.Write(".");
}
while (config.LastUpdateStatus == LastUpdateStatus.InProgress);

await lambdaMethods.UpdateFunctionConfigurationAsync(
    lambdaClient,
    functionName,
    configuration["CalculatorHandler"],
    new Dictionary<string, string> { { "LOG_LEVEL", "DEBUG" } });

do
{
    config = await lambdaMethods.GetFunction(lambdaClient, functionName);
    Console.Write(".");
}
while (config.LastUpdateStatus == LastUpdateStatus.InProgress);

Console.WriteLine();
Console.WriteLine(sepBar);
Console.WriteLine("Now call the updated function...");

// Get two numbers and an action from the user.
value = string.Empty;
do
{
    Console.Write("Enter the first value: ");
    value = Console.ReadLine();
}
while (value == string.Empty);

string? value2;
do
{
    Console.Write("Enter a second value: ");
    value2 = Console.ReadLine();
}
while (value2 == string.Empty);

string? opSelected;

Console.WriteLine("Select the operation to perform:");
Console.WriteLine("\t1. add");
Console.WriteLine("\t2. subtract");
Console.WriteLine("\t3. multiply");
Console.WriteLine("\t4. divide");
Console.WriteLine("Enter the number (1, 2, 3, or 4) of the operation you want to
    perform: ");
do
{
    Console.Write("Your choice? ");
```

```
        opSelected = Console.ReadLine();
    }
    while (opSelected == string.Empty);

    var operation = (opSelected) switch
    {
        "1" => "add",
        "2" => "subtract",
        "3" => "multiply",
        "4" => "divide",
        _ => "add",
    };

    functionParameters = "{" +
        "\"action\": \"\" + operation + "\", " +
        "\"x\": \"\" + value + "\", " +
        "\"y\": \"\" + value2 + "\"" +
    "}";

    answer = await lambdaMethods.InvokeFunctionAsync(lambdaClient, functionName,
        functionParameters);
    Console.WriteLine($"The answer when we {operation} the two numbers is: {answer}.");

    PressEnter();

    // Delete the function created earlier.
    Console.WriteLine(sepBar);
    Console.WriteLine("Delete the AWS Lambda function.");
    var success = await lambdaMethods.DeleteLambdaFunction(lambdaClient, functionName);
    if (success)
    {
        Console.WriteLine($"The {functionName} function was deleted.");
    }
    else
    {
        Console.WriteLine($"Could not remove the function {functionName}");
    }

    // Now delete the IAM role created for use with the functions
    // created by the application.
    success = await lambdaRoleMethods.DeleteLambdaRole(roleName);
    if (success)
    {
        Console.WriteLine("The role has been successfully removed.");
    }
    else
    {
        Console.WriteLine("Couldn't delete the role.");
    }

    Console.WriteLine("The Lambda Scenario is now complete.");
    PressEnter();

    // Displays a formatted list of existing functions returned by the
    // LambdaMethods.ListFunctions.
    void DisplayFunctionList(List<FunctionConfiguration> functions)
    {
        functions.ForEach(functionConfig =>
        {
            Console.WriteLine($"{functionConfig.FunctionName}\t{functionConfig.Description}");
        });
    }

    // Displays an overview of the application.
    void ShowOverview()
```

```
{  
    Console.WriteLine("Welcome to the AWS Lambda Basics Example");  
    Console.WriteLine("Getting started with functions");  
    Console.WriteLine(sepBar);  
    Console.WriteLine("This scenario performs the following operations:");  
    Console.WriteLine("\t 1. Creates an IAM policy that will be used by AWS  
Lambda.");  
    Console.WriteLine("\t 2. Attaches the policy to a new IAM role.");  
    Console.WriteLine("\t 3. Creates an AWS Lambda function.");  
    Console.WriteLine("\t 4. Gets a specific AWS Lambda function.");  
    Console.WriteLine("\t 5. Lists all Lambda functions.");  
    Console.WriteLine("\t 6. Invokes the Lambda function.");  
    Console.WriteLine("\t 7. Updates the Lambda function's code.");  
    Console.WriteLine("\t 8. Updates the Lambda function's configuration.");  
    Console.WriteLine("\t 9. Invokes the updated function.");  
    Console.WriteLine("\t10. Deletes the Lambda function.");  
    Console.WriteLine("\t11. Deletes the IAM role.");  
    PressEnter();  
}  
  
// Wait for the user to press the Enter key.  
void PressEnter()  
{  
    Console.Write("Press <Enter> to continue.");  
    _ = Console.ReadLine();  
    Console.WriteLine();  
}
```

Define a Lambda handler that increments a number.

```
using Amazon.Lambda.Core;  
  
// Assembly attribute to enable the Lambda function's JSON input to be converted  
// into a .NET class.  
[assembly:  
LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer))]  
  
namespace LambdaIncrement;  
  
public class Function  
{  
  
    ///<summary>  
    /// A simple function increments the integer parameter.  
    ///</summary>  
    ///<param name="input">A JSON string containing an action, which must be  
    /// "increment" and a string representing the value to increment.</param>  
    ///<param name="context">The context object passed by Lambda containing  
    /// information about invocation, function, and execution environment.</param>  
    ///<returns>A string representing the incremented value of the parameter.</returns>  
    public int FunctionHandler(Dictionary<string, string> input, ILambdaContext context)  
    {  
        if (input["action"] == "increment")  
        {  
            int inputValue = Convert.ToInt32(input["x"]);  
            return inputValue + 1;  
        }  
        else  
        {  
            return 0;  
        }  
    }  
}
```

```
    }  
}
```

Define a second Lambda handler that performs arithmetic operations.

```
using Amazon.Lambda.Core;  
using System.Diagnostics;  
  
// Assembly attribute to enable the Lambda function's JSON input to be converted  
// into a .NET class.  
[assembly:  
    LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer))]  
  
namespace LambdaCalculator;  
  
public class Function  
{  
  
    /// <summary>  
    /// A simple function that takes two number in string format and performs  
    /// the requested arithmetic function.  
    /// </summary>  
    /// <param name="input">JSON data containing an action, and x and y values.  
    /// Valid actions include: add, subtract, multiply, and divide.</param>  
    /// <param name="context">The context object passed by Lambda containing  
    /// information about invocation, function, and execution environment.</param>  
    /// <returns>A string representing the results of the calculation.</returns>  
    public int FunctionHandler(Dictionary<string, string> input, ILambdaContext  
context)  
    {  
        var action = input["action"];  
        int x = Convert.ToInt32(input["x"]);  
        int y = Convert.ToInt32(input["y"]);  
        int result;  
        switch (action)  
        {  
            case "add":  
                result = x + y;  
                break;  
            case "subtract":  
                result = x - y;  
                break;  
            case "multiply":  
                result = x * y;  
                break;  
            case "divide":  
                if (y == 0)  
                {  
                    Console.Error.WriteLine("Divide by zero error.");  
                    result = 0;  
                }  
                else  
                    result = x / y;  
                break;  
            default:  
                Console.Error.WriteLine($"{action} is not a valid operation.");  
                result = 0;  
                break;  
        }  
        return result;  
    }  
}
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/*
 * Lambda function names appear as:
 *
 * arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
 *
 * To find this value, look at the function in the AWS Management Console.
 *
 * Before running this Java code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example performs the following tasks:
 *
 * 1. Creates an AWS Lambda function.
 * 2. Gets a specific AWS Lambda function.
 * 3. Lists all Lambda functions.
 * 4. Invokes a Lambda function.
 * 5. Updates the Lambda function code and invokes it again.
 * 6. Updates a Lambda function's configuration value.
 * 7. Deletes a Lambda function.
 */

public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");
    public static void main(String[] args) throws InterruptedException {

        final String usage = "\n" +
            "Usage:\n" +
            "      <functionName> <filePath> <role> <handler> <bucketName> <key> \n"
\n" +
            "Where:\n" +
            "      functionName - The name of the Lambda function. \n"+
            "      filePath - The path to the .zip or .jar where the code is located.
\n"+
            "      role - The AWS Identity and Access Management (IAM) service role
that has Lambda permissions. \n"+
    
```

```
"      handler - The fully qualified method name (for example,  
example.Handler::handleRequest). \n"+  
"      bucketName - The Amazon Simple Storage Service (Amazon S3) bucket  
name that contains the .zip or .jar used to update the Lambda function's code.  
\n"+  
"      key - The Amazon S3 key name that represents the .zip or .jar (for  
example, LambdaHello-1.0-SNAPSHOT.jar)." ;  
  
    if (args.length != 6) {  
        System.out.println(usage);  
        System.exit(1);  
    }  
  
    String functionName = args[0];  
    String filePath = args[1];  
    String role = args[2];  
    String handler = args[3];  
    String bucketName = args[4];  
    String key = args[5];  
  
    Region region = Region.US_WEST_2;  
    LambdaClient awsLambda = LambdaClient.builder()  
        .region(region)  
        .credentialsProvider(ProfileCredentialsProvider.create())  
        .build();  
  
    System.out.println(DASHES);  
    System.out.println("Welcome to the AWS Lambda example scenario.");  
    System.out.println(DASHES);  
  
    System.out.println(DASHES);  
    System.out.println("1. Create an AWS Lambda function.");  
    String funArn = createLambdaFunction(awsLambda, functionName, filePath,  
role, handler);  
    System.out.println("The AWS Lambda ARN is "+funArn);  
    System.out.println(DASHES);  
  
    System.out.println(DASHES);  
    System.out.println("2. Get the "+functionName + " AWS Lambda function.");  
    getFunction(awsLambda, functionName);  
    System.out.println(DASHES);  
  
    System.out.println(DASHES);  
    System.out.println("3. List all AWS Lambda functions.");  
    listFunctions(awsLambda);  
    System.out.println(DASHES);  
  
    System.out.println(DASHES);  
    System.out.println("4. Invoke the Lambda function.");  
    System.out.println("*** Sleep for 1 min to get Lambda function ready.");  
    Thread.sleep(60000);  
    invokeFunction(awsLambda, functionName);  
    System.out.println(DASHES);  
  
    System.out.println(DASHES);  
    System.out.println("5. Update the Lambda function code and invoke it  
again.");  
    updateFunctionCode(awsLambda, functionName, bucketName, key);  
    System.out.println("*** Sleep for 1 min to get Lambda function ready.");  
    Thread.sleep(60000);  
    invokeFunction(awsLambda, functionName);  
    System.out.println(DASHES);  
  
    System.out.println(DASHES);  
    System.out.println("6. Update a Lambda function's configuration value.");  
    updateFunctionConfiguration(awsLambda, functionName, handler);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the AWS Lambda function.");
LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The AWS Lambda scenario completed successfully");
System.out.println(DASHES);
awsLambda.close();
}

public static String createLambdaFunction(LambdaClient awsLambda,
                                         String functionName,
                                         String filePath,
                                         String role,
                                         String handler) {

    try {
        LambdaWaiter waiter = awsLambda.waiter();
        InputStream is = new FileInputStream(filePath);
        SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

        FunctionCode code = FunctionCode.builder()
            .zipFile(fileToUpload)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .functionName(functionName)
            .description("Created by the Lambda Java API")
            .code(code)
            .handler(handler)
            .runtime(Runtime.JAVA8)
            .role(role)
            .build();

        // Create a Lambda function using a waiter
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();
        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        return functionResponse.functionArn();

    } catch(LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void getFunction(LambdaClient awsLambda, String functionName) {
    try {
        GetFunctionRequest functionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        GetFunctionResponse response = awsLambda.getFunction(functionRequest);
        System.out.println("The runtime of this Lambda function is "
+response.configuration().runtime());

    } catch(LambdaException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }

    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config: list) {
                System.out.println("The function name is "+config.functionName());
            }
        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void invokeFunction(LambdaClient awsLambda, String functionName)
    {

        InvokeResponse res;
        try {
            // Need a SdkBytes instance for the payload.
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json) ;

            InvokeRequest request = InvokeRequest.builder()
                .functionName(functionName)
                .payload(payload)
                .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String() ;
            System.out.println(value);

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
                .functionName(functionName)
                .publish(true)
                .s3Bucket(bucketName)
                .s3Key(key)
                .build();

            UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest) ;
            GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .build();

            WaiterResponse<GetFunctionConfigurationResponse> waiterResponse =
waiter.waitUntilFunctionUpdated(getFunctionConfigRequest);
        }
    }
}
```

```
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The last modified value is "
+response.lastModified());

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler ){
    try {
        UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
            .functionName(functionName)
            .handler(handler)
            .runtime(Runtime.JAVA11 )
            .build();

        awsLambda.updateFunctionConfiguration(configurationRequest);

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName ) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("The "+functionName +" function was deleted");

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an AWS Identity and Access Management (IAM) role that grants Lambda permission to write to logs.

```
log(`Creating role (${NAME_ROLE_LAMBDA})...`);  
const response = await createRole({  
    AssumeRolePolicyDocument: parseString({  
        Version: "2012-10-17",  
        Statement: [  
            {  
                Effect: "Allow",  
                Principal: {  
                    Service: "lambda.amazonaws.com",  
                },  
                Action: "sts:AssumeRole",  
            },  
        ],  
    }),  
    RoleName: NAME_ROLE_LAMBDA,  
});  
  
const attachRolePolicy = async (roleName, policyArn) => {  
    const client = createClientForDefaultRegion(IAMClient);  
    const command = new AttachRolePolicyCommand({  
        PolicyArn: policyArn, // For example, arn:aws:iam::aws:policy/service-role/  
        AWSLambdaBasicExecutionRole  
        RoleName: roleName, // For example, lambda-basic-execution-role  
    });  
  
    return client.send(command);  
};
```

Create a Lambda function and upload handler code.

```
const createFunction = async (funcName, roleArn) => {  
    const client = createClientForDefaultRegion(LambdaClient);  
    const code = await zip(`${dirname}..functions/${funcName}`);  
  
    const command = new CreateFunctionCommand({  
        Code: { ZipFile: code },  
        FunctionName: funcName,  
        Role: roleArn,  
        Architectures: [Architecture.arm64],  
        Handler: "index.handler", // Required when sending a .zip file  
        PackageType: PackageType.Zip, // Required when sending a .zip file  
        Runtime: Runtime.nodejs16x, // Required when sending a .zip file  
    });  
  
    return client.send(command);  
};
```

Invoke the function with a single parameter and get results.

```
const invoke = async (funcName, payload) => {
```

```
const client = createClientForDefaultRegion(LambdaClient);
const command = new InvokeCommand({
  FunctionName: funcName,
  Payload: JSON.stringify(payload),
  LogType: LogType.Tail,
});

const { Payload, LogResult } = await client.send(command);
const result = Buffer.from(Payload).toString();
const logs = Buffer.from(LogResult, "base64").toString();
return { logs, result };
};
```

Update the function code and configure its Lambda environment with an environment variable.

```
const updateFunctionCode = async (funcName, newFunc) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const code = await zip(`.${dirname}..functions/${newFunc}`);
  const command = new UpdateFunctionCodeCommand({
    ZipFile: code,
    FunctionName: funcName,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};

const updateFunctionConfiguration = async (funcName) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const config = readFileSync(
    `${dirname}..functions/${funcName}/config.json`
  ).toString();
  const command = new UpdateFunctionConfigurationCommand({
    ...JSON.parse(config),
    FunctionName: funcName,
  });
  return client.send(command);
};
```

List the functions for your account.

```
const listFunctions = async () => {
  const client = createClientForDefaultRegion(LambdaClient);
  const command = new ListFunctionsCommand({});

  return client.send(command);
};
```

Delete the IAM role and the Lambda function.

```
const deleteRole = (roleName) => {
  const client = createClientForDefaultRegion(IAMClient);
  const command = new DeleteRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

```
const deleteFunction = (funcName) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const command = new DeleteFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun main(args: Array<String>) {

    val usage = """
        Usage:
            <functionName> <role> <handler> <bucketName> <updatedBucketName> <key>

        Where:
            functionName - The name of the AWS Lambda function.
            role - The AWS Identity and Access Management (IAM) service role that
                  has AWS Lambda permissions.
            handler - The fully qualified method name (for example,
                      example.Handler::handleRequest).
            bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name
                         that contains the ZIP or JAR used for the Lambda function's code.
            updatedBucketName - The Amazon S3 bucket name that contains the .zip
                                or .jar used to update the Lambda function's code.
            key - The Amazon S3 key name that represents the .zip or .jar file (for
                  example, LambdaHello-1.0-SNAPSHOT.jar).
        """

        if (args.size != 6) {
            println(usage)
            exitProcess(1)
        }

        val functionName = args[0]
        val role = args[1]
        val handler = args[2]
        val bucketName = args[3]
        val updatedBucketName = args[4]
        val key = args[5]
```

```
    println("Creating a Lambda function named $functionName.")
    val funArn = createScFunction(functionName, bucketName, key, handler, role)
    println("The AWS Lambda ARN is $funArn")

    // Get a specific Lambda function.
    println("Getting the $functionName AWS Lambda function.")
    getFunction(functionName)

    // List the Lambda functions.
    println("Listing all AWS Lambda functions.")
    listFunctionsSc()

    // Invoke the Lambda function.
    println("**** Invoke the Lambda function.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function code.
    println("**** Update the Lambda function code.")
    updateFunctionCode(functionName, updatedBucketName, key)

    // println("**** Invoke the function again after updating the code.")
    invokeFunctionSc(functionName)

    // Update the AWS Lambda function configuration.
    println("Update the run time of the function.")
    UpdateFunctionConfiguration(functionName, handler)

    // Delete the AWS Lambda function.
    println("Delete the AWS Lambda function.")
    delFunction(functionName)
}

suspend fun createScFunction(
    myFunctionName: String,
    s3BucketName: String,
    myS3Key: String,
    myHandler: String,
    myRole: String
): String {

    val functionCode = FunctionCode {
        s3Bucket = s3BucketName
        s3Key = myS3Key
    }

    val request = CreateFunctionRequest {
        functionName = myFunctionName
        code = functionCode
        description = "Created by the Lambda Kotlin API"
        handler = myHandler
        role = myRole
        runtime = Runtime.Java8
    }

    // Create a Lambda function using a waiter
    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val functionResponse = awsLambda.createFunction(request)
        awsLambda.waitUntilFunctionActive {
            functionName = myFunctionName
        }
        return functionResponse.functionArn.toString()
    }
}

suspend fun getFunction(functionNameVal: String) {
```

```
    val functionRequest = GetFunctionRequest {
        functionName = functionNameVal
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val response = awsLambda.getFunction(functionRequest)
        println("The runtime of this Lambda function is
${response.configuration?.runtime}")
    }
}

suspend fun listFunctionsSc() {

    val request = ListFunctionsRequest {
        maxItems = 10
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val response = awsLambda.listFunctions(request)
        response.functions?.forEach { function ->
            println("The function name is ${function.functionName}")
        }
    }
}

suspend fun invokeFunctionSc(functionNameVal: String) {

    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request = InvokeRequest {
        functionName = functionNameVal
        payload = byteArray
        logType = LogType.Tail
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("The function payload is ${res.payload?.toString(Charsets.UTF_8)}")
    }
}

suspend fun updateFunctionCode(functionNameVal: String?, bucketName: String?, key: String?) {

    val functionCodeRequest = UpdateFunctionCodeRequest {
        functionName = functionNameVal
        publish = true
        s3Bucket = bucketName
        s3Key = key
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val response = awsLambda.updateFunctionCode(functionCodeRequest)
        awsLambda.waitUntilFunctionUpdated {
            functionName = functionNameVal
        }
        println("The last modified value is " + response.lastModified)
    }
}

suspend fun UpdateFunctionConfiguration(functionNameVal: String?, handlerVal: String?) {

    val configurationRequest = UpdateFunctionConfigurationRequest {
        functionName = functionNameVal
    }
}
```

```
        handler = handlerVal
        runtime = Runtime.Java11
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.updateFunctionConfiguration(configurationRequest)
    }
}

suspend fun delFunction(myFunctionName: String) {

    val request = DeleteFunctionRequest {
        functionName = myFunctionName
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.deleteFunction(request)
        println("$myFunctionName was deleted")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace Lambda;

use Aws\S3\S3Client;
use GuzzleHttp\Psr7\Stream;
use Iam\IamService;

class GettingStartedWithLambda
{
    public function run()
    {
        echo("-----\n");
        print("Welcome to the AWS Lambda getting started demo using PHP!\n");
        echo("-----\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();
```

```
$iamService = new IamService();
$s3Client = new S3Client($clientArgs);
$lambdaService = new LambdaService();

echo "First, let's create a role to run our Lambda code.\n";
$roleName = "test-lambda-role-$uniqid";
$rolePolicyDocument = "{$
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Principal\": {
                \"Service\": \"lambda.amazonaws.com\"
            },
            \"Action\": \"sts:AssumeRole\"
        }
    ]
}";
$role = $iamService->createRole($roleName, $rolePolicyDocument);
echo "Created role {$role['RoleName']}.\n";

$iamService->attachRolePolicy(
    $role['RoleName'],
    "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
);
echo "Attached the AWSLambdaBasicExecutionRole to {$role['RoleName']}.\n";

echo "\nNow let's create an S3 bucket and upload our Lambda code there.\n";
$bucketName = "test-example-bucket-$uniqid";
$s3client->createBucket([
    'Bucket' => $bucketName,
]);
echo "Created bucket $bucketName.\n";

$functionName = "doc_example_lambda_$uniqid";
$codeBasic = "lambda_handler_basic.zip";
$handler = "lambda_handler_basic";
$file = file_get_contents($codeBasic);
$s3client->putObject([
    'Bucket' => $bucketName,
    'Key' => $functionName,
    'Body' => $file,
]);
echo "Uploaded the Lambda code.\n";

$createLambdaFunction = $lambdaService->createFunction($functionName,
$role, $bucketName, $handler);
// Wait until the function has finished being created.
do {
    $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
    } while ($getLambdaFunction['Configuration']['State'] == "Pending");
    echo "Created Lambda function {$getLambdaFunction['Configuration']}
['FunctionName'].\n";

sleep(1);

echo "\nOk, let's invoke that Lambda code.\n";
$basicParams = [
    'action' => 'increment',
    'number' => 3,
];
/** @var Stream $invokeFunction */
$invokeFunction = $lambdaService->invoke($functionName, $basicParams)
['Payload'];
$result = json_decode($invokeFunction->getContents())->result;
```

```
echo "After invoking the Lambda code with the input of
{$basicParams['number']} we received $result.\n";

echo "\nSince that's working, let's update the Lambda code.\n";
$codeCalculator = "lambda_handler_calculator.zip";
$handlerCalculator = "lambda_handler_calculator";
echo "First, put the new code into the S3 bucket.\n";
$file = file_get_contents($codeCalculator);
$s3client->putObject([
    'Bucket' => $bucketName,
    'Key' => $functionName,
    'Body' => $file,
]);
echo "New code uploaded.\n";

$lambdaService->updateFunctionCode($functionName, $bucketName,
$functionName);
// Wait for the Lambda code to finish updating.
do {
    $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
} while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
"Successful");
echo "New Lambda code uploaded.\n";

$environment = [
    'Variable' => ['Variables' => ['LOG_LEVEL' => 'DEBUG']],
];
$lambdaService->updateFunctionConfiguration($functionName,
$handlerCalculator, $environment);
do {
    $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
} while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
"Successful");
echo "Lambda code updated with new handler and a LOG_LEVEL of DEBUG for
more information.\n";

echo "Invoke the new code with some new data.\n";
$calculatorParams = [
    'action' => 'plus',
    'x' => 5,
    'y' => 4,
];
$invokeFunction = $lambdaService->invoke($functionName, $calculatorParams,
"Tail");
$result = json_decode($invokeFunction['Payload']->getContents())->result;
echo "Indeed, {$calculatorParams['x']} + {$calculatorParams['y']} does
equal $result.\n";
echo "Here's the extra debug info: ";
echo base64_decode($invokeFunction['LogResult']) . "\n";

echo "\nBut what happens if you try to divide by zero?\n";
$divZeroParams = [
    'action' => 'divide',
    'x' => 5,
    'y' => 0,
];
$invokeFunction = $lambdaService->invoke($functionName, $divZeroParams,
"Tail");
$result = json_decode($invokeFunction['Payload']->getContents())->result;
echo "You get a |$result| result.\n";
echo "And an error message: ";
echo base64_decode($invokeFunction['LogResult']) . "\n";

echo "\nHere's all the Lambda functions you have in this Region:\n";
```

```
$listLambdaFunctions = $lambdaService->listFunctions(5);
$allLambdaFunctions = $listLambdaFunctions['Functions'];
$next = $listLambdaFunctions->get('NextMarker');
while ($next != false) {
    $listLambdaFunctions = $lambdaService->listFunctions(5, $next);
    $next = $listLambdaFunctions->get('NextMarker');
    $allLambdaFunctions = array_merge($allLambdaFunctions,
$listLambdaFunctions['Functions']);
}
foreach ($allLambdaFunctions as $function) {
    echo "{$function['FunctionName']}\\n";
}

echo "\\n\\nAnd don't forget to clean up your data!\\n";

$lambdaService->deleteFunction($functionName);
echo "Deleted Lambda function.\\n";
$iamService->deleteRole($role['RoleName']);
echo "Deleted Role.\\n";
$deleteObjects = $s3client->listObjectsV2([
    'Bucket' => $bucketName,
]);
$deleteObjects = $s3client->deleteObjects([
    'Bucket' => $bucketName,
    'Delete' => [
        'Objects' => $deleteObjects['Contents'],
    ]
]);
echo "Deleted all objects from the S3 bucket.\\n";
$s3client->deleteBucket(['Bucket' => $bucketName]);
echo "Deleted the bucket.\\n";
}
}
```

- For API details, see the following topics in *AWS SDK for PHP API Reference*.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Define a Lambda handler that increments a number.

```
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)
```

```
def lambda_handler(event, context):
    """
        Accepts an action and a single number, performs the specified action on the
        number,
        and returns the result. The only allowable action is 'increment'.

        :param event: The event dict that contains the parameters sent when the
                      function
                           is invoked.
        :param context: The context in which the function is called.
        :return: The result of the action.
    """
    result = None
    action = event.get('action')
    if action == 'increment':
        result = event.get('number', 0) + 1
        logger.info('Calculated result of %s', result)
    else:
        logger.error("%s is not a valid action.", action)

    response = {'result': result}
    return response
```

Define a second Lambda handler that performs arithmetic operations.

```
import logging
import os

logger = logging.getLogger()

# Define a list of Python lambda functions that are called by this AWS Lambda
# function.
ACTIONS = {
    'plus': lambda x, y: x + y,
    'minus': lambda x, y: x - y,
    'times': lambda x, y: x * y,
    'divided-by': lambda x, y: x / y}

def lambda_handler(event, context):
    """
        Accepts an action and two numbers, performs the specified action on the
        numbers,
        and returns the result.

        :param event: The event dict that contains the parameters sent when the
                      function
                           is invoked.
        :param context: The context in which the function is called.
        :return: The result of the specified action.
    """
    # Set the log level based on a variable configured in the Lambda environment.
    logger.setLevel(os.environ.get('LOG_LEVEL', logging.INFO))
    logger.debug('Event: %s', event)

    action = event.get('action')
    func = ACTIONS.get(action)
    x = event.get('x')
    y = event.get('y')
    result = None
    try:
        if func is not None and x is not None and y is not None:
            result = func(x, y)
```

```

        logger.info("%s %s %s is %s", x, action, y, result)
    else:
        logger.error("I can't calculate %s %s %s.", x, action, y)
    except ZeroDivisionError:
        logger.warning("I can't divide %s by 0!", x)

    response = {'result': result}
    return response

```

Create functions that wrap Lambda actions.

```

class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    @staticmethod
    def create_deployment_package(source_file, destination_file):
        """
        Creates a Lambda deployment package in .zip format in an in-memory buffer.
        This buffer can be passed directly to Lambda when creating the function.

        :param source_file: The name of the file that contains the Lambda handler
                            function.
        :param destination_file: The name to give the file when it's deployed to
                                 Lambda.
        :return: The deployment package.
        """
        buffer = io.BytesIO()
        with zipfile.ZipFile(buffer, 'w') as zipped:
            zipped.write(source_file, destination_file)
        buffer.seek(0)
        return buffer.read()

    def get_iam_role(self, iam_role_name):
        """
        Get an AWS Identity and Access Management (IAM) role.

        :param iam_role_name: The name of the role to retrieve.
        :return: The IAM role.
        """
        role = None
        try:
            temp_role = self.iam_resource.Role(iam_role_name)
            temp_role.load()
            role = temp_role
            logger.info("Got IAM role %s", role.name)
        except ClientError as err:
            if err.response['Error']['Code'] == 'NoSuchEntity':
                logger.info("IAM role %s does not exist.", iam_role_name)
            else:
                logger.error(
                    "Couldn't get IAM role %s. Here's why: %s: %s",
                    iam_role_name,
                    err.response['Error']['Code'],
                    err.response['Error']
                )
        raise
        return role

    def create_iam_role_for_lambda(self, iam_role_name):
        """
        Creates an IAM role that grants the Lambda function basic permissions. If a
        role with the specified name already exists, it is used for the demo.

```

```

:param iam_role_name: The name of the role to create.
:return: The role and a value that indicates whether the role is newly
created.
"""
role = self.get_iam_role(iam_role_name)
if role is not None:
    return role, False

lambda_assume_role_policy = {
    'Version': '2012-10-17',
    'Statement': [
        {
            'Effect': 'Allow',
            'Principal': {
                'Service': 'lambda.amazonaws.com'
            },
            'Action': 'sts:AssumeRole'
        }
    ]
}
policy_arn = 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole'

try:
    role = self.iam_resource.create_role(
        RoleName=iam_role_name,
        AssumeRolePolicyDocument=json.dumps(lambda_assume_role_policy))
    logger.info("Created role %s.", role.name)
    role.attach_policy(PolicyArn=policy_arn)
    logger.info("Attached basic execution policy to role %s.", role.name)
except ClientError as error:
    if error.response['Error']['Code'] == 'EntityAlreadyExists':
        role = self.iam_resource.Role(iam_role_name)
        logger.warning("The role %s already exists. Using it.", iam_role_name)
    else:
        logger.exception(
            "Couldn't create role %s or attach policy %s.",
            iam_role_name, policy_arn)
        raise

return role, True

def get_function(self, function_name):
"""
Gets data about a Lambda function.

:param function_name: The name of the function.
:return: The function data.
"""
response = None
try:
    response = self.lambda_client.get_function(FunctionName=function_name)
except ClientError as err:
    if err.response['Error']['Code'] == 'ResourceNotFoundException':
        logger.info("Function %s does not exist.", function_name)
    else:
        logger.error(
            "Couldn't get function %s. Here's why: %s: %s",
            function_name, err.response['Error']['Code'], err.response['Error']
        )
raise
return response

def create_function(self, function_name, handler_name, iam_role,
deployment_package):

```

```
"""
Deploys a Lambda function.

:param function_name: The name of the Lambda function.
:param handler_name: The fully qualified name of the handler function. This
                     must include the file name and the function name.
:param iam_role: The IAM role to use for the function.
:param deployment_package: The deployment package that contains the
                           function
                           code in .zip format.
:return: The Amazon Resource Name (ARN) of the newly created function.
"""

try:
    response = self.lambda_client.create_function(
        FunctionName=function_name,
        Description="AWS Lambda doc example",
        Runtime='python3.8',
        Role=iam_role.arn,
        Handler=handler_name,
        Code={'ZipFile': deployment_package},
        Publish=True)
    function_arn = response['FunctionArn']
    waiter = self.lambda_client.get_waiter('function_active_v2')
    waiter.wait(FunctionName=function_name)
    logger.info("Created function '%s' with ARN: '%s'.",
                function_name, response['FunctionArn'])
except ClientError:
    logger.error("Couldn't create function %s.", function_name)
    raise
else:
    return function_arn

def delete_function(self, function_name):
    """
    Deletes a Lambda function.

    :param function_name: The name of the function to delete.
    """

    try:
        self.lambda_client.delete_function(FunctionName=function_name)
    except ClientError:
        logger.exception("Couldn't delete function %s.", function_name)
        raise

def invoke_function(self, function_name, function_params, get_log=False):
    """
    Invokes a Lambda function.

    :param function_name: The name of the function to invoke.
    :param function_params: The parameters of the function as a dict. This dict
                           is serialized to JSON before it is sent to Lambda.
    :param get_log: When true, the last 4 KB of the execution log are included
                   in
                   the response.
    :return: The response from the function invocation.
    """

    try:
        response = self.lambda_client.invoke(
            FunctionName=function_name,
            Payload=json.dumps(function_params),
            LogType='Tail' if get_log else 'None')
        logger.info("Invoked function %s.", function_name)
    except ClientError:
        logger.exception("Couldn't invoke function %s.", function_name)
        raise
    return response
```

```
def update_function_code(self, function_name, deployment_package):
    """
    Updates the code for a Lambda function by submitting a .zip archive that
    contains
        the code for the function.

    :param function_name: The name of the function to update.
    :param deployment_package: The function code to update, packaged as bytes
    in
        .zip format.
    :return: Data about the update, including the status.
    """
    try:
        response = self.lambda_client.update_function_code(
            FunctionName=function_name, ZipFile=deployment_package)
    except ClientError as err:
        logger.error(
            "Couldn't update function %s. Here's why: %s: %s",
            function_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response

def update_function_configuration(self, function_name, env_vars):
    """
    Updates the environment variables for a Lambda function.

    :param function_name: The name of the function to update.
    :param env_vars: A dict of environment variables to update.
    :return: Data about the update, including the status.
    """
    try:
        response = self.lambda_client.update_function_configuration(
            FunctionName=function_name, Environment={'Variables': env_vars})
    except ClientError as err:
        logger.error(
            "Couldn't update function configuration %s. Here's why: %s: %s",
            function_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response

def list_functions(self):
    """
    Lists the Lambda functions for the current account.
    """
    try:
        funcPaginator = self.lambda_client.getPaginator('list_functions')
        for funcPage in funcPaginator.paginate():
            for func in funcPage['Functions']:
                print(func['FunctionName'])
                desc = func.get('Description')
                if desc:
                    print(f"\t{desc}")
                    print(f"\t{func['Runtime']}: {func['Handler']}")
    except ClientError as err:
        logger.error(
            "Couldn't list functions. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
```

Create a function that runs the scenario.

```
class UpdateFunctionWaiter(CustomWaiter):
    """A custom waiter that waits until a function is successfully updated."""
    def __init__(self, client):
        super().__init__(
            'UpdateSuccess', 'GetFunction',
            'Configuration.LastUpdateStatus',
            {'Successful': WaitState.SUCCESS, 'Failed': WaitState.FAILURE},
            client)

    def wait(self, function_name):
        self._wait(FunctionName=function_name)

def run_scenario(lambda_client, iam_resource, basic_file, calculator_file,
                 lambda_name):
    """
    Runs the scenario.

    :param lambda_client: A Boto3 Lambda client.
    :param iam_resource: A Boto3 IAM resource.
    :param basic_file: The name of the file that contains the basic Lambda handler.
    :param calculator_file: The name of the file that contains the calculator
    Lambda handler.
    :param lambda_name: The name to give resources created for the scenario, such
    as the
                           IAM role and the Lambda function.
    """
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the AWS Lambda getting started with functions demo.")
    print('*'*88)

    wrapper = LambdaWrapper(lambda_client, iam_resource)

    print("Checking for IAM role for Lambda...")
    iam_role, should_wait = wrapper.create_iam_role_for_lambda(lambda_name)
    if should_wait:
        logger.info("Giving AWS time to create resources...")
        wait(10)

    print(f"Looking for function {lambda_name}...")
    function = wrapper.get_function(lambda_name)
    if function is None:
        print("Zipping the Python script into a deployment package...")
        deployment_package = wrapper.create_deployment_package(basic_file,
                                                               f"{lambda_name}.py")
        print(f"...and creating the {lambda_name} Lambda function.")
        wrapper.create_function(
            lambda_name, f'{lambda_name}.lambda_handler', iam_role,
            deployment_package)
    else:
        print(f"Function {lambda_name} already exists.")
    print('*'*88)

    print(f"Let's invoke {lambda_name}. This function increments a number.")
    action_params = {
        'action': 'increment',
        'number': q.ask("Give me a number to increment: ", q.is_int)}
    print(f"Invoking {lambda_name}...")
    response = wrapper.invoke_function(lambda_name, action_params)
    print(f"Incrementing {action_params['number']} resulted in "
          f"[{json.load(response['Payload'])}]")
    print('*'*88)
```

```

print(f"Let's update the function to an arithmetic calculator.")
q.ask("Press Enter when you're ready.")
print("Creating a new deployment package...")
deployment_package = wrapper.create_deployment_package(calculator_file,
f"{{lambda_name}.py}")
print(f"...and updating the {lambda_name} Lambda function.")
update_waiter = UpdateFunctionWaiter(lambda_client)
wrapper.update_function_code(lambda_name, deployment_package)
update_waiter.wait(lambda_name)
print(f"This function uses an environment variable to control logging level.")
print(f"Let's set it to DEBUG to get the most logging.")
wrapper.update_function_configuration(
    lambda_name, {'LOG_LEVEL': logging.getLoggerName(logging.DEBUG)})

actions = ['plus', 'minus', 'times', 'divided-by']
want_invoke = True
while want_invoke:
    print(f"Let's invoke {lambda_name}. You can invoke these actions:")
    for index, action in enumerate(actions):
        print(f"{index + 1}: {action}")
    action_params = {}
    action_index = q.ask(
        "Enter the number of the action you want to take: ",
        q.is_int, q.in_range(1, len(actions)))
    action_params['action'] = actions[action_index - 1]
    print(f"You've chosen to invoke 'x {action_params['action']} y'.")
    action_params['x'] = q.ask("Enter a value for x: ", q.is_int)
    action_params['y'] = q.ask("Enter a value for y: ", q.is_int)
    print(f"Invoking {lambda_name}...")
    response = wrapper.invoke_function(lambda_name, action_params, True)
    print(f"Calculating {action_params['x']} {action_params['action']}")
    action_params['y'] =
        f"resulted in {json.loads(response['Payload'])}"
    q.ask("Press Enter to see the logs from the call.")
    print(base64.b64decode(response['LogResult']).decode())
    want_invoke = q.ask("That was fun. Shall we do it again? (y/n) ",
q.is_yesno)
    print('*88)

    if q.ask("Do you want to list all of the functions in your account? (y/n) ",
q.is_yesno):
        wrapper.list_functions()
    print('*88)

    if q.ask("Ready to delete the function and role? (y/n) ", q.is_yesno):
        for policy in iam_role.attached_policies.all():
            policy.detach_role(RoleName=iam_role.name)
        iam_role.delete()
        print(f"Deleted role {lambda_name}.")
        wrapper.delete_function(lambda_name)
        print(f"Deleted function {lambda_name}.")

    print("\nThanks for watching!")
    print('*88)

if __name__ == '__main__':
    try:
        run_scenario(
            boto3.client('lambda'), boto3.resource('iam'),
'lambda_handler_basic.py',
            'lambda_handler_calculator.py', 'doc_example_lambda_calculator')
    except Exception:
        logging.exception("Something went wrong with the demo!")

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Set up pre-requisite IAM permissions for a Lambda function capable of writing logs.

```
# Get an AWS Identity and Access Management (IAM) role.
#
# @param iam_role_name: The name of the role to retrieve.
# @param action: Whether to create or destroy the IAM apparatus.
# @return: The IAM role.
def manage_iam(iam_role_name, action)
  role_policy = {
    'Version': "2012-10-17",
    'Statement': [
      {
        'Effect': "Allow",
        'Principal': {
          'Service': "lambda.amazonaws.com"
        },
        'Action': "sts:AssumeRole"
      }
    ]
  }
  case action
  when "create"
    role = $iam_client.create_role(
      role_name: iam_role_name,
      assume_role_policy_document: role_policy.to_json
    )
    $iam_client.attach_role_policy(
      {
        policy_arn: "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole",
        role_name: iam_role_name
      }
    )
    $iam_client.wait_until(:role_exists, { role_name: iam_role_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    @logger.debug("Successfully created IAM role: #{role['role']['arn']}")
    @logger.debug("Enforcing a 10-second sleep to allow IAM role to activate
fully.")
    sleep(10)
    return role, role_policy.to_json
  when "destroy"
    $iam_client.detach_role_policy(
      {
        policy_arn: "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole",
        role_name: iam_role_name
      }
    )
    $iam_client.delete_role(
      {
        role_name: iam_role_name
      }
    )
  end
end
```

```
        {
            policy_arn: "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole",
            role_name: iam_role_name
        }
    )
$iam_client.delete_role(
    role_name: iam_role_name
)
@logger.debug("Detached policy & deleted IAM role: #{iam_role_name}")
else
    raise "Incorrect action provided. Must provide 'create' or 'destroy'"
end
rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating role or attaching policy:\n
#{e.message}")
end
```

Define a Lambda handler that increments a number provided as an invocation parameter.

```
require "logger"

# A function that increments a whole number by one (1) and logs the result.
# Requires a manually-provided runtime parameter, 'number', which must be Int
#
# @param event [Hash] Parameters sent when the function is invoked
# @param context [Hash] Methods and properties that provide information
# about the invocation, function, and execution environment.
# @return incremented_number [String] The incremented number.
def lambda_handler(event:, context:)
    logger = Logger.new($stdout)
    log_level = ENV["LOG_LEVEL"]
    logger.level = case log_level
        when "debug"
            Logger::DEBUG
        when "info"
            Logger::INFO
        else
            Logger::ERROR
        end
    logger.debug("This is a debug log message.")
    logger.info("This is an info log message. Code executed successfully!")
    number = event["number"].to_i
    incremented_number = number + 1
    logger.info("You provided #{number.round} and it was incremented to
#{incremented_number.round}")
    incremented_number.round.to_s
end
```

Zip your Lambda function into a deployment package.

```
# Creates a Lambda deployment package in .zip format.
# This zip can be passed directly as a string to Lambda when creating the
function.
#
# @param source_file: The name of the object, without suffix, for the Lambda file
and zip.
# @return: The deployment package.
def create_deployment_package(source_file)
    Dir.chdir(File.dirname(__FILE__))
    if File.exist?("lambda_function.zip")
```

```

    File.delete("lambda_function.zip")
    @logger.debug("Deleting old zip: lambda_function.zip")
end
Zip::File.open("lambda_function.zip", create: true) {
    |zipfile|
    zipfile.add("lambda_function.rb", "#{source_file}.rb")
}
@logger.debug("Zipping #{source_file}.rb into: lambda_function.zip.")
File.read("lambda_function.zip").to_s
rescue StandardError => e
    @logger.error("There was an error creating deployment package:\n #{e.message}")
end

```

Create a new Lambda function.

```

# Deploys a Lambda function.
#
# @param function_name: The name of the Lambda function.
# @param handler_name: The fully qualified name of the handler function. This
#                      must include the file name and the function name.
# @param role_arn: The IAM role to use for the function.
# @param deployment_package: The deployment package that contains the function
#                           code in .zip format.
# @return: The Amazon Resource Name (ARN) of the newly created function.
def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
        role: role_arn.to_s,
        function_name:,
        handler: handler_name,
        runtime: "ruby2.7",
        code: {
            zip_file: deployment_package
        },
        environment: {
            variables: {
                "LOG_LEVEL" => "info"
            }
        }
    })
    @lambda_client.wait_until(:function_active_v2, { function_name: }) do |w|
        w.max_attempts = 5
        w.delay = 5
    end
    response
rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n
#{e.message}")
end

```

Invoke your Lambda function with optional runtime parameters.

```

# Invokes a Lambda function.
# @param function_name [String] The name of the function to invoke.
# @param payload [nil] Payload containing runtime parameters.
# @return [Object] The response from the function invocation.
def invoke_function(function_name, payload = nil)
    params = { function_name: }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)

```

```
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error executing #{function_name}:\n #{e.message}")
end
```

Update your Lambda function's configuration to inject a new environment variable.

```
# Updates the environment variables for a Lambda function.
# @param function_name: The name of the function to update.
# @param log_level: The log level of the function.
# @return: Data about the update, including the status.
def update_function_configuration(function_name, log_level)
  @lambda_client.update_function_configuration({
    function_name:,
    environment: {
      variables: {
        "LOG_LEVEL" => log_level
      }
    }
  })
  @lambda_client.wait_until(:function_updated_v2, { function_name: }) do |w|
    w.max_attempts = 5
    w.delay = 5
  end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating configurations for #{function_name}:
\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n
#{e.message}")
  end
```

Update your Lambda function's code with a different deployment package containing different code.

```
# Updates the code for a Lambda function by submitting a .zip archive that
contains
# the code for the function.

# @param function_name: The name of the function to update.
# @param deployment_package: The function code to update, packaged as bytes in
#                           .zip format.
# @return: Data about the update, including the status.
def update_function_code(function_name, deployment_package)
  @lambda_client.update_function_code(
    function_name:,
    zip_file: deployment_package
  )
  @lambda_client.wait_until(:function_updated_v2, { function_name: }) do |w|
    w.max_attempts = 5
    w.delay = 5
  end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for: #{function_name}:
\n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\n
#{e.message}")
  end
```

List all existing Lambda functions using the built-in paginator.

```
# Lists the Lambda functions for the current account.
def list_functions
  functions = []
  @lambda_client.list_functions.each do |response|
    response["functions"].each do |function|
      functions.append(function["function_name"])
    end
  end
  functions
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error executing #{function_name}:\n #{e.message}")
end
```

Delete a specific Lambda function.

```
# Deletes a Lambda function.
# @param function_name: The name of the function to delete.
def delete_function(function_name)
  print "Deleting function: #{function_name}..."
  @lambda_client.delete_function(
    function_name:
  )
  print "Done!".green
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
end
```

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

SAP ABAP

## SDK for SAP ABAP

### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  "Create an AWS Identity and Access Management (IAM) role that grants Lambda
  permission to write to logs"
  DATA(lv_policy_document) = `{
    &&
    `"Version":"2012-10-17",` &&
    `"Statement": [` &&
    `{` &&
```

```

        ` "Effect": "Allow", ` &&
        ` "Action": [` &&
            ` "sts:AssumeRole" ` &&
        ` ], ` &&
        ` "Principal": {` &&
            ` "Service": [` &&
                ` "lambda.amazonaws.com" ` &&
            ` ]` &&
        ` }` &&
        ` `]` &&
    ` }` .
TRY.
    DATA(lo_create_role_output) = lo_iam->createrole(
        iv_rolename = iv_role_name
        iv_assumerolepolicydocument = lv_policy_document
        iv_description = 'Grant lambda permission to write to logs'
    ).
    MESSAGE 'IAM Role created' TYPE 'I'.
    WAIT UP TO 10 SECONDS.           " Just to make sure IAM role is ready
for use "
    CATCH /aws1/cx_iamentityalrdyexecex.
        MESSAGE 'IAM role already exist' TYPE 'E'.
    CATCH /aws1/cx_iaminvalidinputex.
        MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
    CATCH /aws1/cx_iammalformedplydocex.
        MESSAGE 'Policy document in the request is malformed' TYPE 'E'.
ENDTRY.

TRY.
    lo_iam->attachrolepolicy(
        iv_rolename = iv_role_name
        iv_policyarn = 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole'
    ).
    MESSAGE 'Attached policy to the IAM role' TYPE 'I'.
    CATCH /aws1/cx_iaminvalidinputex.
        MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
    CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'The requested resource entity does not exist' TYPE 'E'.
    CATCH /aws1/cx_iamplynottachableex.
        MESSAGE 'Service role policies can only be attached to the service-
linked role for that service' TYPE 'E'.
    CATCH /aws1/cx_iamunmodableentityex.
        MESSAGE 'Service that depends on the service-linked role is not
modifiable' TYPE 'E'.
ENDTRY.

" Create a Lambda function and upload handler code. "
" Lambda function performs 'increment' action on a number "
TRY.
    lo_lmd->createfunction(
        iv_functionname = iv_function_name
        iv_runtime = `python3.9`
        iv_role = lo_create_role_output->get_role( )->get_arn( )
        iv_handler = iv_handler
        io_code = io_initial_zip_file
        iv_description = 'AWS Lambda code example'
    ).
    MESSAGE 'Lambda function created' TYPE 'I'.
    CATCH /aws1/cx_lmdcodestorageexcdex.
        MESSAGE 'Maximum total code size per account exceeded' TYPE 'E'.
    CATCH /aws1/cx_lmdinvpvalueex.
        MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
    CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist' TYPE 'E'.

```

```

ENDTRY.

" Verify the function is in Active state "
WHILE lo_lmd->getfunction( iv_functionname = iv_function_name )-
>get_configuration( )->ask_state( ) <> 'Active'.
    IF sy-index = 10.
        EXIT.                      " max 10 seconds "
    ENDIF.
    WAIT UP TO 1 SECONDS.
ENDWHILE.

"Invoke the function with a single parameter and get results."
TRY.
    DATA(lv_json) = /aws1/cl_rt_util=>string_to_xstring(
        `{
            `&&
            `'"action": "increment",` &&
            `'"number": 10` &&
        `}`
    ).
    DATA(lo_initial_invoke_output) =  lo_lmd->invoke(
        iv_functionname = iv_function_name
        iv_payload = lv_json
    ).
    ov_initial_invoke_payload = lo_initial_invoke_output->get_payload( ).
    " ov_initial_invoke_payload is returned for testing purpose "
    DATA(lo_writer_json) = cl_sxml_string_writer=>create( type =
if_sxml=>co_xt_json ).
    CALL TRANSFORMATION id SOURCE XML ov_initial_invoke_payload RESULT XML
lo_writer_json.
    DATA(lv_result) = cl_abap_codepage=>convert_from( lo_writer_json-
>get_output( ) ).
        MESSAGE 'Lambda function invoked' TYPE 'I'.
        CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
        CATCH /aws1/cx_lmdinvrequestcontex.
            MESSAGE 'Unable to parse request body as JSON' TYPE 'E'.
        CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist' TYPE 'E'.
        CATCH /aws1/cx_lmdunsuppedmediatyp00.
            MESSAGE 'Invoke request body does not have JSON as its content type'
TYPE 'E'.
    ENDTRY.

    " Update the function code and configure its Lambda environment with an
environment variable. "
    " Lambda function is updated to perform 'decrement' action as well "
TRY.
    lo_lmd->updatefunctioncode(
        iv_functionname = iv_function_name
        iv_zipfile = io_updated_zip_file
    ).
    WAIT UP TO 10 SECONDS.          " Just to make sure update is
completed "
        MESSAGE 'Lambda function code updated' TYPE 'I'.
        CATCH /aws1/cx_lmdcodestorageexcex.
            MESSAGE 'Maximum total code size per account exceeded' TYPE 'E'.
        CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
        CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist' TYPE 'E'.
    ENDTRY.

TRY.
    DATA lt_variables TYPE /aws1/
cl_lmdenvironmentvaria00=>tt_environmentvariables.
    DATA ls_variable LIKE LINE OF lt_variables.

```

```

        ls_variable-key = 'LOG_LEVEL'.
        ls_variable-value = NEW /aws1/cl_lmdenvironmentvaria00( iv_value =
'info' ).
        INSERT ls_variable INTO TABLE lt_variables.

        lo_lmd->updatefunctionconfiguration(
            iv_functionname = iv_function_name
            io_environment = NEW /aws1/cl_lmdenvironment( it_variables =
lt_variables )
        ).
        WAIT UP TO 10 SECONDS.                      " Just to make sure update is
completed "
        MESSAGE 'Lambda function configuration/settings updated' TYPE 'I'.
        CATCH /aws1/cx_lmdinparamvalueex.
        MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
        CATCH /aws1/cx_lmdresourceconflictex.
        MESSAGE 'Rresource already exists or another operation is in progress'
TYPE 'E'.
        CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist' TYPE 'E'.
ENDTRY.

"Invoke the function with new parameters and get results. Display the
execution log that's returned from the invocation."
TRY.
    lv_json = /aws1/cl_rt_util=>string_to_xstring(
        '{` &&
        `'"action": "decrement",` &&
        `'"number": 10` &&
        `'}`
    ).
    DATA(lo_updated_invoke_output) = lo_lmd->invoke(
        iv_functionname = iv_function_name
        iv_payload = lv_json
    ).
    ov_updated_invoke_payload = lo_updated_invoke_output->get_payload( ).
    " ov_updated_invoke_payload is returned for testing purpose "
    lo_writer_json = cl_sxml_string_writer=>create( type =
if_sxml=>co_xt_json ).
    CALL TRANSFORMATION id SOURCE XML ov_updated_invoke_payload RESULT XML
lo_writer_json.
    lv_result = cl_abap_codepage=>convert_from( lo_writer_json-
>get_output( ) ).
    MESSAGE 'Lambda function invoked' TYPE 'I'.
    CATCH /aws1/cx_lmdinparamvalueex.
    MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
    CATCH /aws1/cx_lmdinrequestcontex.
    MESSAGE 'Unable to parse request body as JSON' TYPE 'E'.
    CATCH /aws1/cx_lmdresourcenotfoundex.
    MESSAGE 'The requested resource does not exist' TYPE 'E'.
    CATCH /aws1/cx_lmdunsuppedmediatyp00.
    MESSAGE 'Invoke request body does not have JSON as its content type'
TYPE 'E'.
ENDTRY.

" List the functions for your account. "
TRY.
    DATA(lo_list_output) = lo_lmd->listfunctions( ).
    DATA(lt_functions) = lo_list_output->get_functions( ).
    MESSAGE 'Retrieved list of lambda function(s)' TYPE 'I'.
    CATCH /aws1/cx_lmdinparamvalueex.
    MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
ENDTRY.

" Delete the Lambda function. "
TRY.

```

```
    lo_lmd->deletefunction( iv_functionname = iv_function_name ).  
        MESSAGE 'Lambda function deleted' TYPE 'I'.  
    CATCH /aws1/cx_lmdinvpvalueex.  
        MESSAGE 'The request contains an invalid parameter' TYPE 'E'.  
    CATCH /aws1/cx_lmdresourcenotfoundex.  
        MESSAGE 'The requested resource does not exist' TYPE 'E'.  
    ENDTRY.  
  
    " Detach role policy "  
TRY.  
    lo_iam->detachrolepolicy(  
        iv_rolename = iv_role_name  
        iv_policyarn = 'arn:aws:iam::aws:policy/service-role/  
AWSLambdaBasicExecutionRole'  
    ).  
        MESSAGE 'Detached policy to the IAM role' TYPE 'I'.  
    CATCH /aws1/cx_iaminvalidinputex.  
        MESSAGE 'The request contains an invalid parameter' TYPE 'E'.  
    CATCH /aws1/cx_iamnosuchentityex.  
        MESSAGE 'The requested resource entity does not exist' TYPE 'E'.  
    CATCH /aws1/cx_iamplynottattachableex.  
        MESSAGE 'Service role policies can only be attached to the service-  
linked role for that service' TYPE 'E'.  
    CATCH /aws1/cx_iamunmodableentityex.  
        MESSAGE 'Service that depends on the service-linked role is not  
modifiable' TYPE 'E'.  
    ENDTRY.  
  
    " Delete the IAM role. "  
TRY.  
    lo_iam->deleterole( iv_rolename = iv_role_name ).  
        MESSAGE 'IAM role deleted' TYPE 'I'.  
    CATCH /aws1/cx_iamnosuchentityex.  
        MESSAGE 'The requested resource entity does not exist' TYPE 'E'.  
    CATCH /aws1/cx_iamunmodableentityex.  
        MESSAGE 'Service that depends on the service-linked role is not  
modifiable' TYPE 'E'.  
    ENDTRY.  
  
    CATCH /aws1/cx_rt_service_generic INTO lo_exception.  
        DATA(lv_error) = lo_exception->get_longtext( ).  
        MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## Cross-service examples for Lambda using AWS SDKs

The following code examples show how to use AWS Lambda with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Create an API Gateway REST API to track COVID-19 data \(p. 1283\)](#)
- [Create a lending library REST API \(p. 1283\)](#)
- [Create a messenger application with Step Functions \(p. 1284\)](#)
- [Create a websocket chat application with API Gateway \(p. 1284\)](#)
- [Invoke a Lambda function from a browser \(p. 1285\)](#)
- [Use API Gateway to invoke a Lambda function \(p. 1285\)](#)
- [Use Step Functions to invoke Lambda functions \(p. 1287\)](#)
- [Use scheduled events to invoke a Lambda function \(p. 1288\)](#)

## Create an API Gateway REST API to track COVID-19 data

The following code example shows how to create a REST API that simulates a system to track daily cases of COVID-19 in the United States, using fictional data.

Python

### SDK for Python (Boto3)

Shows how to use AWS Chalice with the AWS SDK for Python (Boto3) to create a serverless REST API that uses Amazon API Gateway, AWS Lambda, and Amazon DynamoDB. The REST API simulates a system that tracks daily cases of COVID-19 in the United States, using fictional data. Learn how to:

- Use AWS Chalice to define routes in Lambda functions that are called to handle REST requests that come through API Gateway.
- Use Lambda functions to retrieve and store data in a DynamoDB table to serve REST requests.
- Define table structure and security role resources in an AWS CloudFormation template.
- Use AWS Chalice and CloudFormation to package and deploy all necessary resources.
- Use CloudFormation to clean up all created resources.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- API Gateway
- AWS CloudFormation
- DynamoDB
- Lambda

## Create a lending library REST API

The following code example shows how to create a lending library where patrons can borrow and return books by using a REST API backed by an Amazon Aurora database.

Python

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with the Amazon Relational Database Service (Amazon RDS) API and AWS Chalice to create a REST API backed by an Amazon Aurora database. The web service is fully serverless and represents a simple lending library where patrons can borrow and return books. Learn how to:

- Create and manage a serverless Aurora database cluster.
- Use AWS Secrets Manager to manage database credentials.
- Implement a data storage layer that uses Amazon RDS to move data into and out of the database.
- Use AWS Chalice to deploy a serverless REST API to Amazon API Gateway and AWS Lambda.
- Use the Requests package to send requests to the web service.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- API Gateway
- Lambda
- Amazon RDS
- Secrets Manager

## Create a messenger application with Step Functions

The following code example shows how to create an AWS Step Functions messenger application that retrieves message records from a database table.

Python

#### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with AWS Step Functions to create a messenger application that retrieves message records from an Amazon DynamoDB table and sends them with Amazon Simple Queue Service (Amazon SQS). The state machine integrates with an AWS Lambda function to scan the database for unsent messages.

- Create a state machine that retrieves and updates message records from an Amazon DynamoDB table.
- Update the state machine definition to also send messages to Amazon Simple Queue Service (Amazon SQS).
- Start and stop state machine runs.
- Connect to Lambda, DynamoDB, and Amazon SQS from a state machine by using service integrations.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Lambda
- Amazon SQS
- Step Functions

## Create a websocket chat application with API Gateway

The following code example shows how to create a chat application that is served by a websocket API built on Amazon API Gateway.

## Python

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with Amazon API Gateway V2 to create a websocket API that integrates with AWS Lambda and Amazon DynamoDB.

- Create a websocket API served by API Gateway.
- Define a Lambda handler that stores connections in DynamoDB and posts messages to other chat participants.
- Connect to the websocket chat application and send messages with the Websockets package.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- API Gateway
- DynamoDB
- Lambda

## Invoke a Lambda function from a browser

The following code example shows how to invoke an AWS Lambda function from a browser.

### JavaScript

#### SDK for JavaScript V2

You can create a browser-based application that uses an AWS Lambda function to update an Amazon DynamoDB table with user selections.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Lambda

#### SDK for JavaScript V3

You can create a browser-based application that uses an AWS Lambda function to update an Amazon DynamoDB table with user selections. This app uses AWS SDK for JavaScript v3.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Lambda

## Use API Gateway to invoke a Lambda function

The following code examples show how to create an AWS Lambda function invoked by Amazon API Gateway.

## Java

### SDK for Java 2.x

Shows how to create an AWS Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create a Lambda function invoked by Amazon API Gateway that scans an Amazon DynamoDB table for work anniversaries and uses Amazon Simple Notification Service (Amazon SNS) to send a text message to your employees that congratulates them at their one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

## JavaScript

### SDK for JavaScript V3

Shows how to create an AWS Lambda function by using the Lambda JavaScript runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create a Lambda function invoked by Amazon API Gateway that scans an Amazon DynamoDB table for work anniversaries and uses Amazon Simple Notification Service (Amazon SNS) to send a text message to your employees that congratulates them at their one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

#### Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

## Python

### SDK for Python (Boto3)

This example shows how to create and use an Amazon API Gateway REST API that targets an AWS Lambda function. The Lambda handler demonstrates how to route based on HTTP methods; how to get data from the query string, header, and body; and how to return a JSON response.

- Deploy a Lambda function.
- Create an API Gateway REST API.
- Create a REST resource that targets the Lambda function.

- Grant permission to let API Gateway invoke the Lambda function.
- Use the Requests package to send requests to the REST API.
- Clean up all resources created during the demo.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- API Gateway
- Lambda

## Use Step Functions to invoke Lambda functions

The following code examples show how to create an AWS Step Functions state machine that invokes AWS Lambda functions in sequence.

Java

#### **SDK for Java 2.x**

Shows how to create an AWS serverless workflow by using AWS Step Functions and the AWS SDK for Java 2.x. Each workflow step is implemented using an AWS Lambda function.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

JavaScript

#### **SDK for JavaScript V3**

Shows how to create an AWS serverless workflow by using AWS Step Functions and the AWS SDK for JavaScript. Each workflow step is implemented using an AWS Lambda function.

Lambda is a compute service that enables you to run code without provisioning or managing servers. Step Functions is a serverless orchestration service that lets you combine Lambda functions and other AWS services to build business-critical applications.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

#### **Services used in this example**

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

## Use scheduled events to invoke a Lambda function

The following code examples show how to create an AWS Lambda function invoked by an Amazon EventBridge scheduled event.

Java

### SDK for Java 2.x

Shows how to create an Amazon EventBridge scheduled event that invokes an AWS Lambda function. Configure EventBridge to use a cron expression to schedule when the Lambda function is invoked. In this example, you create a Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create an app that sends a mobile text message to your employees that congratulates them at the one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

### SDK for JavaScript V3

Shows how to create an Amazon EventBridge scheduled event that invokes an AWS Lambda function. Configure EventBridge to use a cron expression to schedule when the Lambda function is invoked. In this example, you create a Lambda function by using the Lambda JavaScript runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create an app that sends a mobile text message to your employees that congratulates them at the one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

#### Services used in this example

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

### SDK for Python (Boto3)

This example shows how to register an AWS Lambda function as the target of a scheduled Amazon EventBridge event. The Lambda handler writes a friendly message and the full event data to Amazon CloudWatch Logs for later retrieval.

- Deploys a Lambda function.
- Creates an EventBridge scheduled event and makes the Lambda function the target.

- Grants permission to let EventBridge invoke the Lambda function.
- Prints the latest data from CloudWatch Logs to show the result of the scheduled invocations.
- Cleans up all resources created during the demo.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- CloudWatch Logs
- EventBridge
- Lambda

## Code examples for Amazon Lex using AWS SDKs

The following code examples show how to use Amazon Lex with an AWS software development kit (SDK).

#### Code examples

- [Cross-service examples for Amazon Lex using AWS SDKs \(p. 1289\)](#)
  - [Create an Amazon Lex Chatbot within a web application to engage your web site visitors \(p. 1289\)](#)

## Cross-service examples for Amazon Lex using AWS SDKs

The following code examples show how to use Amazon Lex with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

#### Examples

- [Create an Amazon Lex Chatbot within a web application to engage your web site visitors \(p. 1289\)](#)

## Create an Amazon Lex Chatbot within a web application to engage your web site visitors

The following code examples show how to create a Chatbot to engage your web site visitors.

Java

#### SDK for Java 2.x

Shows how to use the Amazon Lex API to create a Chatbot within a web application to engage your web site visitors.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

## JavaScript

### SDK for JavaScript V3

Shows how to use the Amazon Lex API to create a Chatbot within a web application to engage your web site visitors.

For complete source code and instructions on how to set up and run, see the full example [Building an Amazon Lex chatbot](#) in the AWS SDK for JavaScript developer guide.

#### Services used in this example

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

## Code examples for Lookout for Vision using AWS SDKs

The following code examples show how to use Amazon Lookout for Vision with an AWS software development kit (SDK).

#### Code examples

- [Actions for Lookout for Vision using AWS SDKs \(p. 1290\)](#)
  - Create a Lookout for Vision dataset using an AWS SDK (p. 1291)
  - Create a Lookout for Vision model using an AWS SDK (p. 1292)
  - Create a Lookout for Vision project using an AWS SDK (p. 1294)
  - Delete a Lookout for Vision dataset using an AWS SDK (p. 1294)
  - Delete a Lookout for Vision model using an AWS SDK (p. 1295)
  - Delete a Lookout for Vision project using an AWS SDK (p. 1296)
  - Describe a Lookout for Vision dataset using an AWS SDK (p. 1296)
  - Describe a Lookout for Vision model using an AWS SDK (p. 1297)
  - Detect anomalies in an image with a trained Lookout for Vision model using an AWS SDK (p. 1298)
  - List Lookout for Vision models using an AWS SDK (p. 1303)
  - List Lookout for Vision projects using an AWS SDK (p. 1304)
  - Start a Lookout for Vision model using an AWS SDK (p. 1305)
  - Stop a Lookout for Vision model using an AWS SDK (p. 1306)
- [Scenarios for Lookout for Vision using AWS SDKs \(p. 1307\)](#)
  - Create a Lookout for Vision manifest file using an AWS SDK (p. 1307)
  - Create, train, and start a Lookout for Vision model using an AWS SDK (p. 1309)
  - Find a Lookout for Vision project with a specific tag using an AWS SDK (p. 1309)
  - List Lookout for Vision models that are currently hosted using an AWS SDK (p. 1311)

## Actions for Lookout for Vision using AWS SDKs

The following code examples show how to use Amazon Lookout for Vision with AWS SDKs. Each example calls an individual service function.

## Examples

- [Create a Lookout for Vision dataset using an AWS SDK \(p. 1291\)](#)
- [Create a Lookout for Vision model using an AWS SDK \(p. 1292\)](#)
- [Create a Lookout for Vision project using an AWS SDK \(p. 1294\)](#)
- [Delete a Lookout for Vision dataset using an AWS SDK \(p. 1294\)](#)
- [Delete a Lookout for Vision model using an AWS SDK \(p. 1295\)](#)
- [Delete a Lookout for Vision project using an AWS SDK \(p. 1296\)](#)
- [Describe a Lookout for Vision dataset using an AWS SDK \(p. 1296\)](#)
- [Describe a Lookout for Vision model using an AWS SDK \(p. 1297\)](#)
- [Detect anomalies in an image with a trained Lookout for Vision model using an AWS SDK \(p. 1298\)](#)
- [List Lookout for Vision models using an AWS SDK \(p. 1303\)](#)
- [List Lookout for Vision projects using an AWS SDK \(p. 1304\)](#)
- [Start a Lookout for Vision model using an AWS SDK \(p. 1305\)](#)
- [Stop a Lookout for Vision model using an AWS SDK \(p. 1306\)](#)

## Create a Lookout for Vision dataset using an AWS SDK

The following code example shows how to create a Lookout for Vision dataset.

For more information, see [Creating your dataset](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Datasets:

    @staticmethod
    def create_dataset(lookoutvision_client, project_name, manifest_file,
dataset_type):
        """
        Creates a new Lookout for Vision dataset

        :param lookoutvision_client: A Lookout for Vision Boto3 client.
        :param project_name: The name of the project in which you want to
create a dataset.
        :param bucket: The bucket that contains the manifest file.
        :param manifest_file: The path and name of the manifest file.
        :param dataset_type: The type of the dataset (train or test).
        """
        try:

            bucket, key = manifest_file.replace("s3://", "").split("/", 1)
            logger.info("Creating %s dataset type...", dataset_type)
            dataset = {
                "GroundTruthManifest": {"S3Object": {"Bucket": bucket, "Key": key}}
            }
            response = lookoutvision_client.create_dataset(
                ProjectName=project_name,
                DatasetType=dataset_type,
                DatasetSource=dataset,
```

```
        )
    logger.info("Dataset Status: %s",
                response["DatasetMetadata"]["Status"])
    logger.info(
        "Dataset Status Message: %s",
        response["DatasetMetadata"]["StatusMessage"],
    )
    logger.info("Dataset Type: %s",
                response["DatasetMetadata"]["DatasetType"])

    # Wait until either created or failed.
    finished = False
    status = ""
    dataset_description = {}
    while finished is False:
        dataset_description = lookoutvision_client.describe_dataset(
            ProjectName=project_name, DatasetType=dataset_type
        )
        status = dataset_description["DatasetDescription"]["Status"]

        if status == "CREATE_IN_PROGRESS":
            logger.info("Dataset creation in progress...")
            time.sleep(2)
        elif status == "CREATE_COMPLETE":
            logger.info("Dataset created.")
            finished = True
        else:
            logger.info(
                "Dataset creation failed: %s",
                dataset_description["DatasetDescription"]["StatusMessage"]
            )
            finished = True

        if status != "CREATE_COMPLETE":
            message = dataset_description["DatasetDescription"][
                "StatusMessage"]
            logger.exception("Couldn't create dataset: %s", message)
            raise Exception(f"Couldn't create dataset: {message}")

    except ClientError:
        logger.exception("Service error: Couldn't create dataset.")
        raise
```

- For API details, see [CreateDataset](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a Lookout for Vision model using an AWS SDK

The following code example shows how to create a Lookout for Vision model.

For more information, see [Training your model](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Models:
```

```

@staticmethod
def create_model(
    lookoutvision_client, project_name, training_results, tag_key=None,
    tag_key_value=None):
    """
    Creates a version of a Lookout for Vision model.

    :param lookoutvision_client: A Boto3 Lookout for Vision client.
    :param project_name: The name of the project in which you want to create a
                         model.
    :param training_results: The Amazon S3 location where training results are
                           stored.
    :param tag_key: The key for a tag to add to the model.
    :param tag_key_value - A value associated with the tag_key.
    return: The model status and version.
    """
    try:
        logger.info("Training model...")
        output_bucket, output_folder = training_results.replace(
            "s3://", "").split("/", 1)
        output_config = {
            "S3Location": {"Bucket": output_bucket, "Prefix": output_folder}}
        tags = []
        if tag_key is not None:
            tags = [{"Key": tag_key, "Value": tag_key_value}]

        response = lookoutvision_client.create_model(
            ProjectName=project_name, OutputConfig=output_config, Tags=tags)

        logger.info("ARN: %s", response["ModelMetadata"]["ModelArn"])
        logger.info("Version: %s", response["ModelMetadata"]["ModelVersion"])
        logger.info("Started training...")

        print("Training started. Training might take several hours to
complete.")

        # Wait until training completes.
        finished = False
        status = "UNKNOWN"
        while finished is False:
            model_description = lookoutvision_client.describe_model(
                ProjectName=project_name,
                ModelVersion=response["ModelMetadata"]["ModelVersion"])
            status = model_description["ModelDescription"]["Status"]

            if status == "TRAINING":
                logger.info("Model training in progress...")
                time.sleep(600)
                continue

            if status == "TRAINED":
                logger.info("Model was successfully trained.")
            else:
                logger.info(
                    "Model training failed: %s ",
                    model_description["ModelDescription"]["StatusMessage"])
            finished = True
        except ClientError:
            logger.exception("Couldn't train model.")
            raise
    else:
        return status, response["ModelMetadata"]["ModelVersion"]

```

- For API details, see [CreateModel](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a Lookout for Vision project using an AWS SDK

The following code example shows how to create a Lookout for Vision project.

For more information, see [Creating your project](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Projects:

    @staticmethod
    def create_project(lookoutvision_client, project_name):
        """
        Creates a new Lookout for Vision project.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name for the new project.
        :return project_arn: The ARN of the new project.
        """
        try:
            logger.info("Creating project: %s", project_name)
            response =
                lookoutvision_client.create_project(ProjectName=project_name)
            project_arn = response["ProjectMetadata"]["ProjectArn"]
            logger.info("project ARN: %s", project_arn)
        except ClientError:
            logger.exception("Couldn't create project %s.", project_name)
            raise
        else:
            return project_arn
```

- For API details, see [CreateProject in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete a Lookout for Vision dataset using an AWS SDK

The following code example shows how to delete a Lookout for Vision dataset.

For more information, see [Deleting a dataset](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Datasets:

    @staticmethod
    def delete_dataset(lookoutvision_client, project_name, dataset_type):
```

```
"""
Deletes a Lookout for Vision dataset

:param lookoutvision_client: A Boto3 Lookout for Vision client.
:param project_name: The name of the project that contains the dataset that
                     you want to delete.
:param dataset_type: The type (train or test) of the dataset that you
                     want to delete.
"""
try:
    logger.info(
        "Deleting the %s dataset for project %s.", dataset_type,
        project_name)
    lookoutvision_client.delete_dataset(
        ProjectName=project_name, DatasetType=dataset_type)
    logger.info("Dataset deleted.")
except ClientError:
    logger.exception("Service error: Couldn't delete dataset.")
    raise
```

- For API details, see [DeleteDataset](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a Lookout for Vision model using an AWS SDK

The following code example shows how to delete a Lookout for Vision model.

For more information, see [Deleting a model](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Models:

    @staticmethod
    def delete_model(lookoutvision_client, project_name, model_version):
        """
        Deletes a Lookout for Vision model. The model must first be stopped and
        can't
        be in training.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the desired
        model.
        :param model_version: The version of the model that you want to delete.
        """
        try:
            logger.info("Deleting model: %s", model_version)
            lookoutvision_client.delete_model(
                ProjectName=project_name, ModelVersion=model_version)

            model_exists = True
            while model_exists:
                response =
                    lookoutvision_client.list_models(ProjectName=project_name)

            model_exists = False
```

```
for model in response["Models"]:
    if model["ModelVersion"] == model_version:
        model_exists = True

    if model_exists is False:
        logger.info("Model deleted")
    else:
        logger.info("Model is being deleted...")
        time.sleep(2)

    logger.info("Deleted Model: %s", model_version)
except ClientError:
    logger.exception("Couldn't delete model.")
    raise
```

- For API details, see [DeleteModel](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a Lookout for Vision project using an AWS SDK

The following code example shows how to delete a Lookout for Vision project.

For more information, see [Deleting a project](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Projects:

    @staticmethod
    def delete_project(lookoutvision_client, project_name):
        """
        Deletes a Lookout for Vision Model

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that you want to delete.
        """
        try:
            logger.info("Deleting project: %s", project_name)
            response =
                lookoutvision_client.delete_project(ProjectName=project_name)
            logger.info("Deleted project ARN: %s ", response["ProjectArn"])
        except ClientError as err:
            logger.exception("Couldn't delete project %s.", project_name)
            raise
```

- For API details, see [DeleteProject](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a Lookout for Vision dataset using an AWS SDK

The following code example shows how to describe a Lookout for Vision dataset.

For more information, see [Viewing your dataset](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Datasets:

    @staticmethod
    def describe_dataset(lookoutvision_client, project_name, dataset_type):
        """
        Gets information about a Lookout for Vision dataset.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the dataset that you want to describe.
        :param dataset_type: The type (train or test) of the dataset that you want to describe.
        """
        try:
            response = lookoutvision_client.describe_dataset(
                ProjectName=project_name, DatasetType=dataset_type)
            print(f"Name: {response['DatasetDescription']['ProjectName']}")
            print(f"Type: {response['DatasetDescription']['DatasetType']}")
            print(f"Status: {response['DatasetDescription']['Status']}")
            print(
                f"Message: {response['DatasetDescription']['StatusMessage']}")
            print(
                f"Images: {response['DatasetDescription']['ImageStats']['Total']}")
            print(
                f"Labeled: {response['DatasetDescription']['ImageStats']['Labeled']}")
            print(
                f"Normal: {response['DatasetDescription']['ImageStats']['Normal']}")
            print(
                f"Anomaly: {response['DatasetDescription']['ImageStats']['Anomaly']}")
        except ClientError:
            logger.exception("Service error: problem listing datasets.")
            raise
        print("Done.")
```

- For API details, see [DescribeDataset](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a Lookout for Vision model using an AWS SDK

The following code example shows how to describe a Lookout for Vision model.

For more information, see [Viewing your models](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Models:

    @staticmethod
    def describe_model(lookoutvision_client, project_name, model_version):
        """
        Shows the performance metrics for a trained model.

        :param lookoutvision_client: A Boto3 Amazon Lookout for Vision client.
        :param project_name: The name of the project that contains the desired
        model.
        :param model_version: The version of the model.
        """
        response = lookoutvision_client.describe_model(
            ProjectName=project_name, ModelVersion=model_version)
        model_description = response["ModelDescription"]
        print(f"\tModel version: {model_description['ModelVersion']}")
        print(f"\tARN: {model_description['ModelArn']}")
        if "Description" in model_description:
            print(f"\tDescription: {model_description['Description']}")
        print(f"\tStatus: {model_description['Status']}")
        print(f"\tMessage: {model_description['StatusMessage']}")
        print(f"\tCreated: {str(model_description['CreationTimestamp'])}")

        if model_description['Status'] in ("TRAINED", "HOSTED"):
            training_start = model_description["CreationTimestamp"]
            training_end = model_description["EvaluationEndTimestamp"]
            duration = training_end - training_start
            print(f"\tTraining duration: {duration}")

            print("\n\tPerformance metrics\n\t-----")
            print(f"\tRecall: {model_description['Performance']['Recall']}")
            print(f"\tPrecision: {model_description['Performance']['Precision']}")
            print(f"\tF1: {model_description['Performance']['F1Score']}")

        training_output_bucket = model_description["OutputConfig"][
            "S3Location"]["Bucket"]
        prefix = model_description["OutputConfig"]["S3Location"]["Prefix"]
        print(f"\tTraining output: s3://{training_output_bucket}/{prefix}")
```

- For API details, see [DescribeModel in AWS SDK for Python \(Boto3\) API Reference](#).

## Detect anomalies in an image with a trained Lookout for Vision model using an AWS SDK

The following code example shows how to detect anomalies in an image with a trained Lookout for Vision model.

For more information, see [Detecting anomalies in an image](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Inference:
```

```
"""
    Shows how to detect anomalies in an image using a trained Lookout for Vision
model.
"""

@staticmethod
def detect_anomalies(lookoutvision_client, project_name, model_version, photo):
    """
    Calls DetectAnomalies using the supplied project, model version, and image.
    :param lookoutvision_client: A Lookout for Vision Boto3 client.
    :param project: The project that contains the model that you want to use.
    :param model_version: The version of the model that you want to use.
    :param photo: The photo that you want to analyze.
    :return: The DetectAnomalyResult object that contains the analysis results.
    """

    image_type = imghdr.what(photo)
    if image_type == "jpeg":
        content_type = "image/jpeg"
    elif image_type == "png":
        content_type = "image/png"
    else:
        logger.info("Image type not valid for %s", photo)
        raise ValueError(
            f"File format not valid. Supply a jpeg or png format file: {photo}")
    # Get images bytes for call to detect_anomalies.
    with open(photo, "rb") as image:
        response = lookoutvision_client.detect_anomalies(
            ProjectName=project_name,
            ContentType=content_type,
            Body=image.read(),
            ModelVersion=model_version)

    return response['DetectAnomalyResult']

@staticmethod
def download_from_s3(s3_resource, photo):
    """
    Downloads an image from an S3 bucket.

    :param s3_resource: A Boto3 Amazon S3 resource.
    :param photo: The Amazon S3 path of a photo to download.
    :return: The local path to the downloaded file.
    """

    try:
        bucket, key = photo.replace("s3://", "").split("/", 1)
        local_file = os.path.basename(photo)
    except ValueError:
        logger.exception("Couldn't get S3 info for %s", photo)
        raise

    try:
        logger.info("Downloading %s", photo)
        s3_resource.Bucket(bucket).download_file(key, local_file)
    except ClientError:
        logger.exception("Couldn't download %s from S3.", photo)
        raise

    return local_file

@staticmethod
def reject_on_classification(image, prediction, confidence_limit):
    """
```

```

    Returns True if the anomaly confidence is greater than or equal to
    the supplied confidence limit.
    :param image: The name of the image file that was analyzed.
    :param prediction: The DetectAnomalyResult object returned from
DetectAnomalies.
    :param confidence_limit: The minimum acceptable confidence (float 0 - 1).
    :return: True if the error condition indicates an anomaly, otherwise False.
    """

    reject = False

    logger.info("Checking classification for %s", image)

    if prediction['IsAnomalous'] and prediction['Confidence'] >=
confidence_limit:
        reject = True
        reject_info=(f"Rejected: Anomaly confidence
({prediction['Confidence']:.2%}) is greater"
                     f" than limit ({confidence_limit:.2%})")
        logger.info("%s", reject_info)

    if not reject:
        logger.info("No anomalies found.")
    return reject

@staticmethod
def reject_on_anomaly_types(image, prediction, confidence_limit,
anomaly_types_limit):
    """
    Checks if the number of anomaly types is greater than the anomaly types
    limit and if the prediction confidence is greater than the confidence
    limit.
    :param image: The name of the image file that was analyzed.
    :param prediction: The DetectAnomalyResult object returned from
DetectAnomalies.
    :param confidence: The minimum acceptable confidence (float 0 - 1).
    :param anomaly_types_limit: The maximum number of allowable anomaly types
(int).
    :return: True if the error condition indicates an anomaly, otherwise False.
    """

    logger.info("Checking number of anomaly types for %s",image)

    reject = False

    if prediction['IsAnomalous'] and prediction['Confidence'] >=
confidence_limit:

        anomaly_types = {anomaly['Name'] for anomaly in
prediction['Anomalies']\
                         if anomaly['Name'] != 'background'}

        if len (anomaly_types) > anomaly_types_limit:
            reject = True
            reject_info = (f"Rejected: Anomaly confidence
({prediction['Confidence']:.2%}) "
                           f"is greater than limit ({confidence_limit:.2%}) and "
                           f"the number of anomaly types ({len(anomaly_types)-1}) is "
                           f"greater than the limit ({anomaly_types_limit})")

            logger.info("%s", reject_info)

    if not reject:
        logger.info("No anomalies found.")
    return reject

```

```

@staticmethod
def reject_on_coverage(image, prediction, confidence_limit, anomaly_label,
                      coverage_limit):
    """
        Checks if the coverage area of an anomaly is greater than the coverage
        limit and if
            the prediction confidence is greater than the confidence limit.
        :param image: The name of the image file that was analyzed.
        :param prediction: The DetectAnomalyResult object returned from
                           DetectAnomalies.
        :param confidence_limit: The minimum acceptable confidence (float 0-1).
        :param anomaly_label: The anomaly label for the type of anomaly that you want to
                              check.
        :param coverage_limit: The maximum acceptable percentage coverage of an anomaly
                              (float 0-1).
        :return: True if the error condition indicates an anomaly, otherwise False.
    """

    reject = False

    logger.info("Checking coverage for %s", image)

    if prediction['IsAnomalous'] and prediction['Confidence'] >=
    confidence_limit:
        for anomaly in prediction['Anomalies']:
            if (anomaly['Name'] == anomaly_label and
                anomaly['PixelAnomaly']['TotalPercentageArea'] >
                (coverage_limit)):
                reject = True
                reject_info=f"Rejected: Anomaly confidence
                ({prediction['Confidence']:.2%}) "
                f"is greater than limit ({confidence_limit:.2%}) and
                {anomaly['Name']} "
                f"coverage ({anomaly['PixelAnomaly']
                ['TotalPercentageArea']:.2%}) "
                f"is greater than limit ({coverage_limit:.2%})"

                logger.info("%s", reject_info)

    if not reject:
        logger.info("No anomalies found.")

    return reject

@staticmethod
def analyze_image(lookoutvision_client, image, config):
    """
        Analyzes an image with an Amazon Lookout for Vision model. Also
        runs a series of checks to determine if the contents of an image
        should be rejected.
        :param lookoutvision_client: A Lookout for Vision Boto3 client.
        param image: A local image that you want to analyze.
        param config: Configuration information for the model and reject
                      limits.
    """

    project = config['project']
    model_version = config['model_version']
    confidence_limit = config['confidence_limit']
    coverage_limit = config['coverage_limit']
    anomaly_types_limit = config['anomaly_types_limit']
    anomaly_label = config['anomaly_label']

    # Get analysis results.
    print(f"Analyzing {image}.")

```

```
prediction = Inference.detect_anomalies(
    lookoutvision_client, project, model_version, image)

anomalies = []

reject = Inference.reject_on_classification(
    image, prediction, confidence_limit)

if reject:
    anomalies.append("Classification: An anomaly was found.")

reject = Inference.reject_on_coverage(
    image, prediction, confidence_limit, anomaly_label, coverage_limit)

if reject:
    anomalies.append("Coverage: Anomaly coverage too high.")

reject = Inference.reject_on_anomaly_types(
    image, prediction, confidence_limit, anomaly_types_limit)

if reject:
    anomalies.append(
        "Anomaly type count: Too many anomaly types found.")
    print()

if len(anomalies) > 0:
    print(f"Anomalies found in {image}")
    for anomaly in anomalies:
        print(f"{anomaly}")
else:
    print(f"No anomalies found in {image}")


def main():
    """
    Detects anomalies in an image file.
    """
    try:
        logging.basicConfig(level=logging.INFO,
                            format="%(levelname)s: %(message)s")

        parser = argparse.ArgumentParser(
            description="Find anomalies with Amazon Lookout for Vision.")
        parser.add_argument(
            "image",
            help="The file that you want to analyze. Supply a local file path or a"
        "                                "path to an S3 object.")
        parser.add_argument(
            "config", help=("The configuration JSON file to use. "
                           "See https://github.com/awsdocs/aws-doc-sdk-examples/blob/main/"
                           "python/example_code/lookoutvision/README.md"))

        args = parser.parse_args()

        lookoutvision_client = boto3.client("lookoutvision")
        s3_resource = boto3.resource('s3')

        # Get configuration information.
        with open(args.config, encoding="utf-8") as config_file:
            config = json.load(config_file)

        # Download image if located in S3 bucket.
        if args.image.startswith("s3://"):
```

```
        image = Inference.download_from_s3(s3_resource, args.image)
    else:
        image = args.image

    Inference.analyze_image(lookoutvision_client, image, config)

    # Delete image, if downloaded from S3 bucket.
    if args.image.startswith("s3://"):
        os.remove(image)

except ClientError as err:
    print(f"Service error: {err.response['Error']['Message']}")
except FileNotFoundError as err:
    print(f"The supplied file couldn't be found: {err.filename}.")
except ValueError as err:
    print(f"A value error occurred: {err}.")
else:
    print("\nSuccessfully completed analysis.")

if __name__ == "__main__":
    main()
```

- For API details, see [DetectAnomalies](#) in *AWS SDK for Python (Boto3) API Reference*.

## List Lookout for Vision models using an AWS SDK

The following code example shows how to list Lookout for Vision models.

For more information, see [Viewing your models](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Models:

    @staticmethod
    def describe_models(lookoutvision_client, project_name):
        """
        Gets information about all models in a Lookout for Vision project.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that you want to use.
        """
        try:
            response = lookoutvision_client.list_models(ProjectName=project_name)
            print("Project: " + project_name)
            for model in response["Models"]:
                Models.describe_model(
                    lookoutvision_client, project_name, model["ModelVersion"])
                print()
            print("Done...")
        except ClientError:
            logger.exception("Couldn't list models.")
            raise
```

- For API details, see [ListModels](#) in *AWS SDK for Python (Boto3) API Reference*.

## List Lookout for Vision projects using an AWS SDK

The following code example shows how to list Lookout for Vision projects.

For more information, see [Viewing your projects](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Projects:

    @staticmethod
    def list_projects(lookoutvision_client):
        """
        Lists information about the projects that are in in your AWS account
        and in the current AWS Region.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        """
        try:
            response = lookoutvision_client.list_projects()
            for project in response["Projects"]:
                print("Project: " + project["ProjectName"])
                print("\tARN: " + project["ProjectArn"])
                print("\tCreated: " + str(["CreationTimestamp"]))
                print("Datasets")
                project_description = lookoutvision_client.describe_project(
                    ProjectName=project["ProjectName"])
                if not project_description["ProjectDescription"]["Datasets"]:
                    print("\tNo datasets")
                else:
                    for dataset in project_description["ProjectDescription"][
                        "Datasets"]:
                        print(f"\t\ttype: {dataset['DatasetType']}")
                        print(f"\t\tStatus: {dataset['StatusMessage']}")

                        print("Models")
                        response_models = lookoutvision_client.list_models(
                            ProjectName=project["ProjectName"])
                        if not response_models["Models"]:
                            print("\tNo models")
                        else:
                            for model in response_models["Models"]:
                                Models.describe_model(
                                    lookoutvision_client, project["ProjectName"],
                                    model["ModelVersion"])

                                print("-----\n")
                print("Done!")
        except ClientError:
            logger.exception("Problem listing projects.")
            raise
```

- For API details, see [ListProjects in AWS SDK for Python \(Boto3\) API Reference](#).

## Start a Lookout for Vision model using an AWS SDK

The following code example shows how to start a Lookout for Vision model.

For more information, see [Starting your model](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Hosting:

    @staticmethod
    def start_model(
        lookoutvision_client, project_name, model_version,
        min_inference_units):
        """
        Starts the hosting of a Lookout for Vision model.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the version of
        the
                           model that you want to start hosting.
        :param model_version: The version of the model that you want to start
        hosting.
        :param min_inference_units: The number of inference units to use for
        hosting.
        """
        try:
            logger.info(
                "Starting model version %s for project %s", model_version,
                project_name)
            lookoutvision_client.start_model(
                ProjectName=project_name,
                ModelVersion=model_version,
                MinInferenceUnits=min_inference_units)
            print("Starting hosting...")

            status = ""
            finished = False

            # Wait until hosted or failed.
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name, ModelVersion=model_version)
                status = model_description["ModelDescription"]["Status"]

                if status == "STARTING_HOSTING":
                    logger.info("Host starting in progress...")
                    time.sleep(10)
                    continue

                if status == "HOSTED":
                    logger.info("Model is hosted and ready for use.")
```

```
        finished = True
        continue

        logger.info("Model hosting failed and the model can't be used.")
        finished = True

        if status != "HOSTED":
            logger.error("Error hosting model: %s", status)
            raise Exception(f"Error hosting model: {status}")
    except ClientError:
        logger.exception("Couldn't host model.")
        raise
```

- For API details, see [StartModel in AWS SDK for Python \(Boto3\) API Reference](#).

## Stop a Lookout for Vision model using an AWS SDK

The following code example shows how to stop a Lookout for Vision model.

For more information, see [Stopping your model](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Hosting:

    @staticmethod
    def stop_model(lookoutvision_client, project_name, model_version):
        """
        Stops a running Lookout for Vision Model.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the version of
                             the model that you want to stop hosting.
        :param model_version: The version of the model that you want to stop
                             hosting.
        """
        try:
            logger.info("Stopping model version %s for %s", model_version,
                       project_name)
            response = lookoutvision_client.stop_model(
                ProjectName=project_name, ModelVersion=model_version)
            logger.info("Stopping hosting...")

            status = response["Status"]
            finished = False

            # Wait until stopped or failed.
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name, ModelVersion=model_version)
                status = model_description["ModelDescription"]["Status"]

                if status == "STOPPING_HOSTING":
                    logger.info("Host stopping in progress...")
                    time.sleep(10)
```

```
        continue

    if status == "TRAINED":
        logger.info("Model is no longer hosted.")
        finished = True
        continue

    logger.info("Failed to stop model: %s ", status)
    finished = True

    if status != "TRAINED":
        logger.error("Error stopping model: %s", status)
        raise Exception(f"Error stopping model: {status}")
    except ClientError:
        logger.exception("Couldn't stop hosting model.")
        raise
```

- For API details, see [StopModel in AWS SDK for Python \(Boto3\) API Reference](#).

## Scenarios for Lookout for Vision using AWS SDKs

The following code examples show how to use Amazon Lookout for Vision with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Create a Lookout for Vision manifest file using an AWS SDK \(p. 1307\)](#)
- [Create, train, and start a Lookout for Vision model using an AWS SDK \(p. 1309\)](#)
- [Find a Lookout for Vision project with a specific tag using an AWS SDK \(p. 1309\)](#)
- [List Lookout for Vision models that are currently hosted using an AWS SDK \(p. 1311\)](#)

## Create a Lookout for Vision manifest file using an AWS SDK

The following code example shows how to create a Lookout for Vision manifest file and upload it to Amazon S3.

For more information, see [Creating a manifest file](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Datasets:

    @staticmethod
    def create_manifest_file_s3(s3_resource, image_s3_path, manifest_s3_path):
        """
        Creates a manifest file and uploads to Amazon S3.

        :param s3_resource: A Boto3 Amazon S3 resource.
        :param image_s3_path: The Amazon S3 path to the images referenced by the
                            manifest file. The images must be in an Amazon S3
                            bucket
        """
```

```
with the following folder structure.  
    s3://doc-example-bucket/<train or test>/  
        normal/  
        anomaly/  
Place normal images in the normal folder and  
anomalous  
    images in the anomaly folder.  
:param manifest_s3_path: The Amazon S3 location in which to store the  
created  
    manifest file.  
    """  
output_manifest_file = "temp.manifest"  
try:  
  
    # Current date and time in manifest file format.  
    dttm = datetime.now().strftime("%Y-%m-%dT%H:%M:%S.%f")  
  
    # Get bucket and folder from image and manifest file paths.  
    bucket, prefix = image_s3_path.replace("s3://", "").split("/", 1)  
    if prefix[-1] != '/':  
        prefix += '/'  
    manifest_bucket, manifest_prefix = manifest_s3_path.replace(  
        "s3://", "").split("/", 1)  
  
    with open(output_manifest_file, "w") as mfile:  
        logger.info("Creating manifest file")  
        src_bucket = s3_resource.Bucket(bucket)  
  
        # Create JSON lines for anomalous images.  
        for obj in src_bucket.objects.filter(  
            Prefix=prefix + "anomaly/", Delimiter="/"):  
            image_path = f"s3://{src_bucket.name}/{obj.key}"  
            manifest = Datasets.create_json_line(  
                image_path, "anomaly", dttm)  
            mfile.write(json.dumps(manifest) + "\n")  
  
        # Create json lines for normal images.  
        for obj in src_bucket.objects.filter(  
            Prefix=prefix + "normal/", Delimiter="/"):  
            image_path = f"s3://{src_bucket.name}/{obj.key}"  
            manifest = Datasets.create_json_line(  
                image_path, "normal", dttm)  
            mfile.write(json.dumps(manifest) + "\n")  
  
        logger.info("Uploading manifest file to %s", manifest_s3_path)  
        s3_resource.Bucket(manifest_bucket).upload_file(  
            output_manifest_file, manifest_prefix)  
    except ClientError:  
        logger.exception("Error uploading manifest.")  
        raise  
    except Exception:  
        logger.exception("Error uploading manifest.")  
        raise  
    else:  
        logger.info("Completed manifest file creation and upload.")  
    finally:  
        try:  
            os.remove(output_manifest_file)  
        except FileNotFoundError:  
            pass  
  
@staticmethod  
def create_json_line(image, class_name, dttm):  
    """  
Creates a single JSON line for an image.
```

```
:param image: The S3 location for the image.
:param class_name: The class of the image (normal or anomaly)
:param dttm: The date and time that the JSON is created.
"""

label = 0
if class_name == "normal":
    label = 0
elif class_name == "anomaly":
    label = 1
else:
    logger.error("Unexpected label value: %s for %s", label, image)
    raise Exception(f"Unexpected label value: {label} for {image}")

manifest = {
    "source-ref": image,
    "anomaly-label": label,
    "anomaly-label-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/anomaly-label",
        "class-name": class_name,
        "human-annotated": "yes",
        "creation-date": dttm,
        "type": "groundtruth/image-classification",
    },
}
return manifest
```

## Create, train, and start a Lookout for Vision model using an AWS SDK

The following code example shows how to create, train, and start a Lookout for Vision model.

Python

### SDK for Python (Boto3)

Creates and optionally starts an Amazon Lookout for Vision model using command line arguments. The example code creates a new project, training dataset, optional test dataset, and model. After model training is completed, you can use a provided script to try your model with an image.

This example requires a set of images to train its model. You can find example circuit board images on GitHub that you can use for training and testing. For details on how to copy these images to an Amazon Simple Storage Service (Amazon S3) bucket, see [Prepare example images](#).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Lookout for Vision

## Find a Lookout for Vision project with a specific tag using an AWS SDK

The following code example shows how to find a Lookout for Vision project with a specific tag.

For more information, see [Tagging models](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import logging
import argparse
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def find_tag(tags, key, value):
    """
    Finds a tag in the supplied list of tags.

    :param tags: A list of tags associated with a Lookout for Vision model.
    :param key: The tag to search for.
    :param value: The tag key value to search for.
    :return: True if the tag value exists, otherwise False.
    """

    found = False
    for tag in tags:
        if key == tag["Key"]:
            logger.info("\t\tMatch found for tag: %s value: %s.", key, value)
            found = True
            break
    return found

def find_tag_in_projects(lookoutvision_client, key, value):
    """
    Finds Lookout for Vision models tagged with the supplied key and value.

    :param lookoutvision_client: A Boto3 Lookout for Vision client.
    :param key: The tag key to find.
    :param value: The value of the tag that you want to find.
    :return: A list of matching model versions (and model projects) that were found.
    """

    try:
        found_tags = []
        found = False
        projects = lookoutvision_client.list_projects()
        # Iterate through each project and models within a project.
        for project in projects["Projects"]:
            logger.info("Searching project: %s ...", project["ProjectName"])

            response_models = lookoutvision_client.list_models(
                ProjectName=project["ProjectName"])
            for model in response_models["Models"]:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project["ProjectName"],
                    ModelVersion=model["ModelVersion"])
                tags = lookoutvision_client.list_tags_for_resource(
                    ResourceArn=model_description["ModelArn"])

                for tag in tags["Tags"]:
                    if key == tag["Key"] and value == tag["Value"]:
                        found_tags.append(model)

        if found:
            logger.info("Found %d matching models.", len(found_tags))
            return found_tags
        else:
            logger.info("No matching models found for tag: %s value: %s.", key, value)
            return []
    except ClientError as e:
        logger.error("An error occurred: %s", e)
        raise
```

```
logger.info(
    "\tSearching model: %s for tag: %s value: %s.",
    model_description["ModelDescription"]["ModelArn"], key, value)
if find_tag(tags["Tags"], key, value) is True:
    found = True
    logger.info(
        "\t\tMATCH: Project: %s: model version %s",
        project["ProjectName"],
        model_description["ModelDescription"]["ModelVersion"])
    found_tags.append({
        "Project": project["ProjectName"],
        "ModelVersion":
            model_description["ModelDescription"]["ModelVersion"]})
if found is False:
    logger.info("No match for tag %s with value %s.", key, value)
except ClientError:
    logger.exception("Problem finding tags.")
    raise
else:
    return found_tags

def main():
    logging.basicConfig(level=logging.INFO, format"%(levelname)s: %(message)s")
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
    parser.add_argument("tag", help="The tag that you want to find.")
    parser.add_argument("value", help="The tag value that you want to find.")
    args = parser.parse_args()
    key = args.tag
    value = args.value
    lookoutvision_client = boto3.client("lookoutvision")

    print(f"Searching your models for tag: {key} with value: {value}.")
    tagged_models = find_tag_in_projects(lookoutvision_client, key, value)

    print("Matched models\n-----")
    if len(tagged_models) > 0:
        for model in tagged_models:
            print(f"Project: {model['Project']}. model version: {model['ModelVersion']}")
    else:
        print("No matches found.")

if __name__ == "__main__":
    main()
```

## List Lookout for Vision models that are currently hosted using an AWS SDK

The following code example shows how to list Lookout for Vision models that are currently hosted.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Hosting:

    @staticmethod
    def list_hosted(lookoutvision_client):
        """
        Displays a list of models in your account that are currently hosted.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        """
        try:
            response = lookoutvision_client.list_projects()
            hosted = 0
            print("Hosted models\n-----")

            for project in response["Projects"]:
                response_models = lookoutvision_client.list_models(
                    ProjectName=project["ProjectName"])

                for model in response_models["Models"]:
                    model_description = lookoutvision_client.describe_model(
                        ProjectName=project["ProjectName"],
                        ModelVersion=model["ModelVersion"])

                    if model_description["ModelDescription"]["Status"] == "HOSTED":
                        print(
                            f"Project: {project['ProjectName']} Model version: "
                            f"{model['ModelVersion']}")
                        hosted += 1
            print(f"{hosted} model(s) hosted")
        except ClientError:
            logger.exception("Problem listing hosted models.")
            raise
```

## Code examples for MediaLive using AWS SDKs

The following code examples show how to use AWS Elemental MediaLive with an AWS software development kit (SDK).

### Code examples

- [Actions for MediaLive using AWS SDKs \(p. 1312\)](#)
- [List your MediaLive inputs using an AWS SDK \(p. 1312\)](#)

## Actions for MediaLive using AWS SDKs

The following code examples show how to use AWS Elemental MediaLive with AWS SDKs. Each example calls an individual service function.

### Examples

- [List your MediaLive inputs using an AWS SDK \(p. 1312\)](#)

## List your MediaLive inputs using an AWS SDK

The following code example shows how to list your MediaLive inputs.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List your MediaLive input names and ARNs in the Region.

```
async fn show_inputs(client: &Client) -> Result<(), Error> {
    let input_list = client.list_inputs().send().await?;

    for i in input_list.inputs().unwrap_or_default() {
        let input_arn = i.arn().unwrap_or_default();
        let input_name = i.name().unwrap_or_default();

        println!("Input Name : {}", input_name);
        println!("Input ARN : {}", input_arn);
        println!();
    }

    Ok(())
}
```

- For API details, see [ListInputs](#) in *AWS SDK for Rust API reference*.

## Code examples for MediaPackage using AWS SDKs

The following code examples show how to use AWS Elemental MediaPackage with an AWS software development kit (SDK).

### Code examples

- [Actions for MediaPackage using AWS SDKs \(p. 1313\)](#)
  - [List your MediaPackage channels using an AWS SDK \(p. 1313\)](#)
  - [List your MediaPackage origin endpoints using an AWS SDK \(p. 1314\)](#)

## Actions for MediaPackage using AWS SDKs

The following code examples show how to use AWS Elemental MediaPackage with AWS SDKs. Each example calls an individual service function.

### Examples

- [List your MediaPackage channels using an AWS SDK \(p. 1313\)](#)
- [List your MediaPackage origin endpoints using an AWS SDK \(p. 1314\)](#)

## List your MediaPackage channels using an AWS SDK

The following code example shows how to list your MediaPackage channels.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List channel ARNs and descriptions.

```
async fn show_channels(client: &Client) -> Result<(), Error> {
    let list_channels = client.list_channels().send().await?;

    println!("Channels:");

    for c in list_channels.channels().unwrap_or_default() {
        let description = c.description().unwrap_or_default();
        let arn = c.arn().unwrap_or_default();

        println!("  Description : {}", description);
        println!("  ARN :          {}", arn);
        println!();
    }

    Ok(())
}
```

- For API details, see [ListChannels](#) in *AWS SDK for Rust API reference*.

## List your MediaPackage origin endpoints using an AWS SDK

The following code example shows how to list your MediaPackage origin endpoints.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List your endpoint descriptions and URLs.

```
async fn show_endpoints(client: &Client) -> Result<(), Error> {
    let or_endpoints = client.list_origin_endpoints().send().await?;

    println!("Endpoints:");

    for e in or_endpoints.origin_endpoints().unwrap_or_default() {
        let endpoint_url = e.url().unwrap_or_default();
        let endpoint_description = e.description().unwrap_or_default();
        println!("  Description: {}", endpoint_description);
    }
}
```

```
    println!(" URL :      {}", endpoint_url);
}
Ok(())
}
```

- For API details, see [ListOriginEndpoints](#) in *AWS SDK for Rust API reference*.

## Code examples for Organizations using AWS SDKs

The following code examples show how to use AWS Organizations with an AWS software development kit (SDK).

### Code examples

- [Actions for Organizations using AWS SDKs \(p. 1315\)](#)
  - [Attach an Organizations policy to a target using an AWS SDK \(p. 1316\)](#)
  - [Create an Organizations policy using an AWS SDK \(p. 1317\)](#)
  - [Create an Organizations account using an AWS SDK \(p. 1319\)](#)
  - [Create an Organizations organization using an AWS SDK \(p. 1319\)](#)
  - [Create an Organizations organizational unit using an AWS SDK \(p. 1320\)](#)
  - [Delete an Organizations policy using an AWS SDK \(p. 1321\)](#)
  - [Delete an Organizations organization using an AWS SDK \(p. 1323\)](#)
  - [Delete an Organizations organizational unit using an AWS SDK \(p. 1323\)](#)
  - [Describe an Organizations policy using an AWS SDK \(p. 1324\)](#)
  - [Detach an Organizations policy from a target using an AWS SDK \(p. 1325\)](#)
  - [List accounts for an Organizations organization using an AWS SDK \(p. 1326\)](#)
  - [List Organizations organizational units using an AWS SDK \(p. 1327\)](#)
  - [List Organizations policies using an AWS SDK \(p. 1329\)](#)

## Actions for Organizations using AWS SDKs

The following code examples show how to use AWS Organizations with AWS SDKs. Each example calls an individual service function.

### Examples

- [Attach an Organizations policy to a target using an AWS SDK \(p. 1316\)](#)
- [Create an Organizations policy using an AWS SDK \(p. 1317\)](#)
- [Create an Organizations account using an AWS SDK \(p. 1319\)](#)
- [Create an Organizations organization using an AWS SDK \(p. 1319\)](#)
- [Create an Organizations organizational unit using an AWS SDK \(p. 1320\)](#)
- [Delete an Organizations policy using an AWS SDK \(p. 1321\)](#)
- [Delete an Organizations organization using an AWS SDK \(p. 1323\)](#)
- [Delete an Organizations organizational unit using an AWS SDK \(p. 1323\)](#)
- [Describe an Organizations policy using an AWS SDK \(p. 1324\)](#)
- [Detach an Organizations policy from a target using an AWS SDK \(p. 1325\)](#)

- [List accounts for an Organizations organization using an AWS SDK \(p. 1326\)](#)
- [List Organizations organizational units using an AWS SDK \(p. 1327\)](#)
- [List Organizations policies using an AWS SDK \(p. 1329\)](#)

## Attach an Organizations policy to a target using an AWS SDK

The following code examples show how to attach an Organizations policy to a target.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to attach an AWS Organizations policy to an organization,
/// an organizational unit, or an account.
/// </summary>
public class AttachPolicy
{
    /// <summary>
    /// Initializes the Organizations client object and then calls the
    /// AttachPolicyAsync method to attach the policy to the root
    /// organization.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
        var policyId = "p-00000000";
        var targetId = "r-0000";

        var request = new AttachPolicyRequest
        {
            PolicyId = policyId,
            TargetId = targetId,
        };

        var response = await client.AttachPolicyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully attached Policy ID {policyId} to
Target ID: {targetId}.");
        }
        else
        {
            Console.WriteLine("Was not successful in attaching the policy.");
        }
    }
}
```

- For API details, see [AttachPolicy](#) in [AWS SDK for .NET API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def attach_policy(policy_id, target_id, orgs_client):
    """
    Attaches a policy to a target. The target is an organization root, account, or
    organizational unit.

    :param policy_id: The ID of the policy to attach.
    :param target_id: The ID of the resources to attach the policy to.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.attach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Attached policy %s to target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't attach policy %s to target %s.", policy_id, target_id)
        raise
```

- For API details, see [AttachPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an Organizations policy using an AWS SDK

The following code examples show how to create an Organizations policy.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

///<summary>
/// Creates a new AWS Organizations Policy. The example was created
/// using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
///</summary>
class CreatePolicy
{
    ///<summary>
    /// Initializes the AWS Organizations client object, uses it to
    /// create a new Organizations Policy, and then displays information
    /// about the newly created Policy.
    ///</summary>
    static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
        var policyContent = "{" +
```

```
"    \"Version\": \"2012-10-17\", " +
" \"Statement\" : [{" +
"   \"Action\" : ["s3:*"], " +
"   \"Effect\" : \"Allow\", " +
"   \"Resource\" : \"*\" +
"}]" +
"}";

try
{
    var response = await client.CreatePolicyAsync(new
CreatePolicyRequest
{
    Content = policyContent,
    Description = "Enables admins of attached accounts to delegate
all Amazon S3 permissions",
    Name = "AllowAllS3Actions",
    Type = "SERVICE_CONTROL_POLICY",
});

    Policy policy = response.Policy;
    Console.WriteLine($"{policy.PolicySummary.Name} has the following
content: {policy.Content}");
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
}
```

- For API details, see [CreatePolicy](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_policy(name, description, content, policy_type, orgs_client):
    """
Creates a policy.

:param name: The name of the policy.
:param description: The description of the policy.
:param content: The policy content as a dict. This is converted to JSON before
it is sent to AWS. The specific format depends on the policy
type.
:param policy_type: The type of the policy.
:param orgs_client: The Boto3 Organizations client.
:return: The newly created policy.
"""
try:
    response = orgs_client.create_policy(
        Name=name, Description=description, Content=json.dumps(content),
        Type=policy_type)
    logger.info("Created policy %s.", name)
except ClientError:
    logger.exception("Couldn't create policy %s.", name)
```

```
        raise
else:
    return policy
```

- For API details, see [CreatePolicy in AWS SDK for Python \(Boto3\) API Reference](#).

## Create an Organizations account using an AWS SDK

The following code example shows how to create an Organizations account.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

///<summary>
/// Creates a new AWS Organizations account. The example was created
/// using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
class CreateAccount
{
    ///<summary>
    /// Initializes an Organizations client object and uses it to create
    /// the new account with the name specified in accountName.
    /// </summary>
    static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
        var accountName = "ExampleAccount";
        var email = "someone@example.com";

        var request = new CreateAccountRequest
        {
            AccountName = accountName,
            Email = email,
        };

        var response = await client.CreateAccountAsync(request);
        var status = response.CreateAccountStatus;

        Console.WriteLine($"The status of {status.AccountName} is
{status.State}.");
    }
}
```

- For API details, see [CreateAccount in AWS SDK for .NET API Reference](#).

## Create an Organizations organization using an AWS SDK

The following code example shows how to create an Organizations organization.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Creates an organization in AWS Organizations. The example was created
/// using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class CreateOrganization
{
    /// <summary>
    /// Creates an Organizations client object and then uses it to create
    /// a new organization with the default user as the administrator, and
    /// then displays information about the new organization.
    /// </summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var response = await client.CreateOrganizationAsync(new
CreateOrganizationRequest
        {
            FeatureSet = "ALL",
        });

        Organization newOrg = response.Organization;

        Console.WriteLine($"Organization: {newOrg.Id} Main Account:
{newOrg.MasterAccountId}");
    }
}
```

- For API details, see [CreateOrganization](#) in *AWS SDK for .NET API Reference*.

## Create an Organizations organizational unit using an AWS SDK

The following code example shows how to create an Organizations organizational unit.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;
```

```
/// <summary>
/// Creates a new organizational unit in AWS Organizations. The example was
/// created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class CreateOrganizationalUnit
{
    /// <summary>
    /// Initializes an Organizations client object and then uses it to call
    /// the CreateOrganizationalUnit method. If the call succeeds, it
    /// displays information about the new organizational unit.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var orgUnitName = "ProductDevelopmentUnit";

        var request = new CreateOrganizationalUnitRequest
        {
            Name = orgUnitName,
            ParentId = "r-0000",
        };

        var response = await client.CreateOrganizationalUnitAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully created organizational unit:
{orgUnitName}.");
            Console.WriteLine($"Organizational unit {orgUnitName} Details");
            Console.WriteLine($"ARN: {response.OrganizationalUnit.Arn} Id:
{response.OrganizationalUnit.Id}");
        }
        else
        {
            Console.WriteLine("Could not create new organizational unit.");
        }
    }
}
```

- For API details, see [CreateOrganizationalUnit](#) in *AWS SDK for .NET API Reference*.

## Delete an Organizations policy using an AWS SDK

The following code examples show how to delete an Organizations policy.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
```

```
/// Deletes an existing AWS Organizations policy. This example was
/// created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DeletePolicy
{
    /// <summary>
    /// Initializes the Organizations client object and then uses it to
    /// delete the policy with the specified policyId.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var policyId = "p-00000000";

        var request = new DeletePolicyRequest
        {
            PolicyId = policyId,
        };

        var response = await client.DeletePolicyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully deleted Policy: {policyId}.");
        }
        else
        {
            Console.WriteLine($"Could not delete Policy: {policyId}.");
        }
    }
}
```

- For API details, see [DeletePolicy in AWS SDK for .NET API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_policy(policy_id, orgs_client):
    """
    Deletes a policy.

    :param policy_id: The ID of the policy to delete.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.delete_policy(PolicyId=policy_id)
        logger.info("Deleted policy %s.", policy_id)
    except ClientError:
        logger.exception(
            "Couldn't delete policy %s.", policy_id)
        raise
```

- For API details, see [DeletePolicy in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete an Organizations organization using an AWS SDK

The following code example shows how to delete an Organizations organization.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

///<summary>
/// Shows how to delete an existing organization using the AWS
/// Organizations Service. This example was created using the AWS SDK for
/// .NET version 3.7 and .NET Core 5.0.
///</summary>
public class DeleteOrganization
{
    ///<summary>
    /// Initializes the Organizations client and then calls
    /// DeleteOrganizationAsync to delete the organization.
    ///</summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var response = await client.DeleteOrganizationAsync(new
DeleteOrganizationRequest());

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("Successfully deleted organization.");
        }
        else
        {
            Console.WriteLine("Could not delete organization.");
        }
    }
}
```

- For API details, see [DeleteOrganization](#) in *AWS SDK for .NET API Reference*.

## Delete an Organizations organizational unit using an AWS SDK

The following code example shows how to delete an Organizations organizational unit.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to delete an existing AWS Organizations organizational unit.
/// This example was created using the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
/// </summary>
public class DeleteOrganizationalUnit
{
    /// <summary>
    /// Initializes the Organizations client object and calls
    /// DeleteOrganizationalUnitAsync to delete the organizational unit
    /// with the selected ID.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var orgUnitId = "ou-0000-00000000";

        var request = new DeleteOrganizationalUnitRequest
        {
            OrganizationalUnitId = orgUnitId,
        };

        var response = await client.DeleteOrganizationalUnitAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully deleted the organizational unit
with ID: {orgUnitId}.");
        }
        else
        {
            Console.WriteLine($"Could not delete the organizational unit with
ID: {orgUnitId}.");
        }
    }
}
```

- For API details, see [DeleteOrganizationalUnit](#) in *AWS SDK for .NET API Reference*.

## Describe an Organizations policy using an AWS SDK

The following code example shows how to describe an Organizations policy.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def describe_policy(policy_id, orgs_client):
    """
```

```
Describes a policy.

:param policy_id: The ID of the policy to describe.
:param orgs_client: The Boto3 Organizations client.
:return: The description of the policy.
"""
try:
    response = orgs_client.describe_policy(PolicyId=policy_id)
    policy = response['Policy']
    logger.info("Got policy %s.", policy_id)
except ClientError:
    logger.exception("Couldn't get policy %s.", policy_id)
    raise
else:
    return policy
```

- For API details, see [DescribePolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detach an Organizations policy from a target using an AWS SDK

The following code examples show how to detach an Organizations policy from a target.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to detach a policy from an AWS Organizations organization,
/// organizational unit, or account. The example was creating using the
/// AWS SDK for .NET 3.7 and .NET Core 5.0.
/// </summary>
public class DetachPolicy
{
    /// <summary>
    /// Initializes the Organizations client object and uses it to call
    /// DetachPolicyAsync to detach the policy.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var policyId = "p-00000000";
        var targetId = "r-0000";

        var request = new DetachPolicyRequest
        {
            PolicyId = policyId,
            TargetId = targetId,
        };

        var response = await client.DetachPolicyAsync(request);
```

```
        if (response.StatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"Successfully detached policy with Policy Id:
{policyId}.");
    }
    else
    {
        Console.WriteLine("Could not detach the policy.");
    }
}
```

- For API details, see [DetachPolicy](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def detach_policy(policy_id, target_id, orgs_client):
    """
    Detaches a policy from a target.

    :param policy_id: The ID of the policy to detach.
    :param target_id: The ID of the resource where the policy is currently
        attached.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.detach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Detached policy %s from target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from target %s.", policy_id, target_id)
        raise
```

- For API details, see [DetachPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## List accounts for an Organizations organization using an AWS SDK

The following code example shows how to list accounts for an Organizations organization.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
```

```
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

///<summary>
/// Uses the AWS Organizations service to list the accounts associated
/// with the default account. The example was created using the AWS SDK for
/// .NET and .NET Core 5.0.
///</summary>
class ListAccounts
{
    ///<summary>
    /// Creates the Organizations client and then calls its
    /// ListAccountsAsync method.
    ///</summary>
    static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var request = new ListAccountsRequest
        {
            MaxResults = 5,
        };

        var response = new ListAccountsResponse();
        try
        {
            do
            {
                response = await client.ListAccountsAsync(request);
                response.Accounts.ForEach(a => DisplayAccounts(a));
                if (response.NextToken is not null)
                {
                    request.NextToken = response.NextToken;
                }
            } while (response.NextToken is not null);
        }
        catch (AWSOrganizationsNotInUseException ex)
        {
            Console.WriteLine(ex.Message);
        }
    }

    ///<summary>
    /// Displays information about an Organizations account.
    ///</summary>
    ///<param name="account">An Organizations account for which to display
    /// information on the console.</param>
    private static void DisplayAccounts(Account account)
    {
        string accountInfo = $"{account.Id} {account.Name}\t{account.Status}";
        Console.WriteLine(accountInfo);
    }
}
```

- For API details, see [ListAccounts in AWS SDK for .NET API Reference](#).

## List Organizations organizational units using an AWS SDK

The following code example shows how to list Organizations organizational units.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Lists the AWS Organizations organizational units that belong to an
/// organization. The example was created using the AWS SDK for .NET
/// version 3.7 and .NET Core 5.0.
/// </summary>
public class ListOrganizationalUnitsForParent
{
    /// <summary>
    /// Initializes the Organizations client object and then uses it to
    /// call the ListOrganizationalUnitsForParentAsync method to retrieve
    /// the list of organizational units.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var parentId = "r-0000";

        var request = new ListOrganizationalUnitsForParentRequest
        {
            ParentId = parentId,
            MaxResults = 5,
        };

        var response = new ListOrganizationalUnitsForParentResponse();
        try
        {
            do
            {
                response = await
client.ListOrganizationalUnitsForParentAsync(request);
                response.OrganizationalUnits.ForEach(u =>
DisplayOrganizationalUnit(u));
                if (response.NextToken is not null)
                {
                    request.NextToken = response.NextToken;
                }
            } while (response.NextToken is not null);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }

    /// <summary>
    /// Displays information about an Organizations organizational unit.
    /// </summary>
    /// <param name="unit">The OrganizationalUnit for which to display
```

```
/// information.</param>
public static void DisplayOrganizationalUnit(OrganizationalUnit unit)
{
    string accountInfo = $"{unit.Id} {unit.Name}\t{unit.ArN}";

    Console.WriteLine(accountInfo);
}
```

- For API details, see [ListOrganizationalUnitsForParent](#) in *AWS SDK for .NET API Reference*.

## List Organizations policies using an AWS SDK

The following code examples show how to list Organizations policies.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to list the AWS Organizations policies associated with an
/// organization. The example was created with the AWS SDK for .NET version
/// 3.7 and .NET Core 5.0.
/// </summary>
public class ListPolicies
{
    /// <summary>
    /// Initializes an Organizations client object, and then calls its
    /// ListPoliciesAsync method.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        // The value for the Filter parameter is required and must be
        // one of the following:
        //    AISERVICES_OPT_OUT_POLICY
        //    BACKUP_POLICY
        //    SERVICE_CONTROL_POLICY
        //    TAG_POLICY
        var request = new ListPoliciesRequest
        {
            Filter = "SERVICE_CONTROL_POLICY",
            MaxResults = 5,
        };

        var response = new ListPoliciesResponse();
        try
        {
            do
```

```
{  
    response = await client.ListPoliciesAsync(request);  
    response.Policies.ForEach(p => DisplayPolicies(p));  
    if (response.NextToken is not null)  
    {  
        request.NextToken = response.NextToken;  
    }  
} while (response.NextToken is not null);  
}  
catch (AWSOrganizationsNotInUseException ex)  
{  
    Console.WriteLine(ex.Message);  
}  
}  
  
/// <summary>  
/// Displays information about the Organizations policies associated  
/// with an organization.  
/// </summary>  
/// <param name="policy">An Organizations policy summary to display  
/// information on the console.</param>  
private static void DisplayPolicies(PolicySummary policy)  
{  
    string policyInfo = $"{policy.Id} {policy.Name}\t{policy.Description}";  
  
    Console.WriteLine(policyInfo);  
}  
}
```

- For API details, see [ListPolicies](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_policies(policy_filter, orgs_client):  
    """  
    Lists the policies for the account, limited to the specified filter.  
  
    :param policy_filter: The kind of policies to return.  
    :param orgs_client: The Boto3 Organizations client.  
    :return: The list of policies found.  
    """  
    try:  
        response = orgs_client.list_policies(Filter=policy_filter)  
        policies = response['Policies']  
        logger.info("Found %s %s policies.", len(policies), policy_filter)  
    except ClientError:  
        logger.exception("Couldn't get %s policies.", policy_filter)  
        raise  
    else:  
        return policies
```

- For API details, see [ListPolicies](#) in *AWS SDK for Python (Boto3) API Reference*.

# Code examples for Amazon Personalize using AWS SDKs

The following code examples show how to use Amazon Personalize with an AWS software development kit (SDK).

## Code examples

- [Actions for Amazon Personalize using AWS SDKs \(p. 1331\)](#)
  - [Create a batch inference job with Amazon Personalize using an AWS SDK \(p. 1332\)](#)
  - [Create a batch segment job with Amazon Personalize using an AWS SDK \(p. 1335\)](#)
  - [Create a campaign with Amazon Personalize using an AWS SDK \(p. 1336\)](#)
  - [Create a dataset with Amazon Personalize using an AWS SDK \(p. 1337\)](#)
  - [Create a dataset export job with Amazon Personalize using an AWS SDK \(p. 1338\)](#)
  - [Create a dataset group with Amazon Personalize using an AWS SDK \(p. 1340\)](#)
  - [Create a dataset import job with Amazon Personalize using an AWS SDK \(p. 1342\)](#)
  - [Create a domain schema with Amazon Personalize using an AWS SDK \(p. 1344\)](#)
  - [Create a filter with Amazon Personalize using an AWS SDK \(p. 1346\)](#)
  - [Create a recommender with Amazon Personalize using an AWS SDK \(p. 1347\)](#)
  - [Create a schema with Amazon Personalize using an AWS SDK \(p. 1349\)](#)
  - [Create a solution with Amazon Personalize using an AWS SDK \(p. 1350\)](#)
  - [Create a solution version with Amazon Personalize using an AWS SDK \(p. 1351\)](#)
  - [Create an event tracker with Amazon Personalize using an AWS SDK \(p. 1354\)](#)
  - [Delete a campaign in Amazon Personalize using an AWS SDK \(p. 1355\)](#)
  - [Delete a solution in Amazon Personalize using an AWS SDK \(p. 1356\)](#)
  - [Delete an event tracker in Amazon Personalize using an AWS SDK \(p. 1356\)](#)
  - [Describe a campaign in Amazon Personalize using an AWS SDK \(p. 1357\)](#)
  - [Describe a recipe in Amazon Personalize using an AWS SDK \(p. 1358\)](#)
  - [Describe a solution in Amazon Personalize using an AWS SDK \(p. 1358\)](#)
  - [Incrementally import an item into Amazon Personalize Events using an AWS SDK \(p. 1359\)](#)
  - [List campaigns in Amazon Personalize using an AWS SDK \(p. 1360\)](#)
  - [List dataset groups in Amazon Personalize using an AWS SDK \(p. 1360\)](#)
  - [List recipes in Amazon Personalize using an AWS SDK \(p. 1361\)](#)
  - [List solutions in Amazon Personalize using an AWS SDK \(p. 1362\)](#)
  - [Update a campaign in Amazon Personalize using an AWS SDK \(p. 1362\)](#)

## Actions for Amazon Personalize using AWS SDKs

The following code examples show how to use Amazon Personalize with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create a batch inference job with Amazon Personalize using an AWS SDK \(p. 1332\)](#)
- [Create a batch segment job with Amazon Personalize using an AWS SDK \(p. 1335\)](#)
- [Create a campaign with Amazon Personalize using an AWS SDK \(p. 1336\)](#)
- [Create a dataset with Amazon Personalize using an AWS SDK \(p. 1337\)](#)
- [Create a dataset export job with Amazon Personalize using an AWS SDK \(p. 1338\)](#)

- Create a dataset group with Amazon Personalize using an AWS SDK (p. 1340)
- Create a dataset import job with Amazon Personalize using an AWS SDK (p. 1342)
- Create a domain schema with Amazon Personalize using an AWS SDK (p. 1344)
- Create a filter with Amazon Personalize using an AWS SDK (p. 1346)
- Create a recommender with Amazon Personalize using an AWS SDK (p. 1347)
- Create a schema with Amazon Personalize using an AWS SDK (p. 1349)
- Create a solution with Amazon Personalize using an AWS SDK (p. 1350)
- Create a solution version with Amazon Personalize using an AWS SDK (p. 1351)
- Create an event tracker with Amazon Personalize using an AWS SDK (p. 1354)
- Delete a campaign in Amazon Personalize using an AWS SDK (p. 1355)
- Delete a solution in Amazon Personalize using an AWS SDK (p. 1356)
- Delete an event tracker in Amazon Personalize using an AWS SDK (p. 1356)
- Describe a campaign in Amazon Personalize using an AWS SDK (p. 1357)
- Describe a recipe in Amazon Personalize using an AWS SDK (p. 1358)
- Describe a solution in Amazon Personalize using an AWS SDK (p. 1358)
- Incrementally import an item into Amazon Personalize Events using an AWS SDK (p. 1359)
- List campaigns in Amazon Personalize using an AWS SDK (p. 1360)
- List dataset groups in Amazon Personalize using an AWS SDK (p. 1360)
- List recipes in Amazon Personalize using an AWS SDK (p. 1361)
- List solutions in Amazon Personalize using an AWS SDK (p. 1362)
- Update a campaign in Amazon Personalize using an AWS SDK (p. 1362)

## Create a batch inference job with Amazon Personalize using an AWS SDK

The following code examples show how to create a Amazon Personalize batch interface job.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                                         String
solutionVersionArn,
                                         String jobName,
String
s3InputDataSourcePath,
                                         String
s3DataDestinationPath,
                                         String roleArn,
String
explorationWeight,
                                         String
explorationItemAgeCutOff) {
    long waitInMilliseconds = 60 * 1000;
    String status;
```

```

String batchInferenceJobArn;

try {

    // Set up data input and output parameters.
    S3DataConfig inputSource = S3DataConfig.builder()
        .path(s3InputDataSourcePath)
        .build();

    S3DataConfig outputDestination = S3DataConfig.builder()
        .path(s3DataDestinationPath)
        .build();

    BatchInferenceJobInput jobInput = BatchInferenceJobInput.builder()
        .s3DataSource(inputSource)
        .build();

    BatchInferenceJobOutput jobOutputLocation =
    BatchInferenceJobOutput.builder()
        .s3DataDestination(outputDestination)
        .build();

    // Optional code to build the User-Personalization specific item
    exploration config.
    HashMap<String, String> explorationConfig = new HashMap<>();
    explorationConfig.put("explorationWeight", explorationWeight);
    explorationConfig.put("explorationItemAgeCutOff",
    explorationItemAgeCutOff);

    BatchInferenceJobConfig jobConfig = BatchInferenceJobConfig.builder()
        .itemExplorationConfig(explorationConfig)
        .build();

    // End optional User-Personalization recipe specific code.

    CreateBatchInferenceJobRequest createBatchInferenceJobRequest =
CreateBatchInferenceJobRequest.builder()
    .solutionVersionArn(solutionVersionArn)
    .jobInput(jobInput)
    .jobOutput(jobOutputLocation)
    .jobName(jobName)
    .roleArn(roleArn)
    .batchInferenceJobConfig(jobConfig) // Optional
    .build();

    batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
    .batchInferenceJobArn();

    DescribeBatchInferenceJobRequest describeBatchInferenceJobRequest =
DescribeBatchInferenceJobRequest.builder()
    .batchInferenceJobArn(batchInferenceJobArn)
    .build();

    long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;
    while (Instant.now().getEpochSecond() < maxTime) {

        BatchInferenceJob batchInferenceJob = personalizeClient
            .describeBatchInferenceJob(describeBatchInferenceJobRequest)
            .batchInferenceJob();

        status = batchInferenceJob.status();
        System.out.println("Batch inference job status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {

```

```
        break;
    }
    try {
        Thread.sleep(waitInMilliseconds);
    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
    }
}
return batchInferenceJobArn;

} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}
```

- For API details, see [CreateBatchInferenceJob](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchInferenceJobCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch inference job's parameters.

export const createBatchInferenceJobParam = {
  jobName: 'JOB_NAME',
  jobInput: { /* required */
    s3DataSource: { /* required */
      path: 'INPUT_PATH', /* required */
      // kmsKeyArn: 'INPUT_KMS_KEY_ARN' /* optional */
    }
  },
  jobOutput: { /* required */
    s3DataDestination: { /* required */
      path: 'OUTPUT_PATH', /* required */
      // kmsKeyArn: 'OUTPUT_KMS_KEY_ARN' /* optional */
    }
  },
  roleArn: 'ROLE_ARN', /* required */
  solutionVersionArn: 'SOLUTION_VERSION_ARN', /* required */
  numResults: 20 /* optional integer*/
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
      CreateBatchInferenceJobCommand(createBatchInferenceJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
```

```
        console.log("Error", err);
    }
};

run();
```

- For API details, see [CreateBatchInferenceJob](#) in *AWS SDK for JavaScript API Reference*.

## Create a batch segment job with Amazon Personalize using an AWS SDK

The following code example shows how to create a Amazon Personalize batch segment job.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchSegmentJobCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch segment job's parameters.

export const createBatchSegmentJobParam = {
  jobName: 'NAME',
  jobInput: { /* required */
    s3DataSource: { /* required */
      path: 'INPUT_PATH', /* required */
      // kmsKeyArn: 'INPUT_KMS_KEY_ARN' /* optional */
    }
  },
  jobOutput: { /* required */
    s3DataDestination: { /* required */
      path: 'OUTPUT_PATH', /* required */
      // kmsKeyArn: 'OUTPUT_KMS_KEY_ARN' /* optional */
    }
  },
  roleArn: 'ROLE_ARN', /* required */
  solutionVersionArn: 'SOLUTION_VERSION_ARN', /* required */
  numResults: 20 /* optional */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateBatchSegmentJobCommand(createBatchSegmentJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
```

```
run();
```

- For API details, see [CreateBatchSegmentJob](#) in *AWS SDK for JavaScript API Reference*.

## Create a campaign with Amazon Personalize using an AWS SDK

The following code examples show how to create a Amazon Personalize campaign.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createPersonalCompaign(PersonalizeClient personalizeClient,
String solutionVersionArn, String name) {

    try {
        CreateCampaignRequest createCampaignRequest =
CreateCampaignRequest.builder()
            .minProvisionedTPS(1)
            .solutionVersionArn(solutionVersionArn)
            .name(name)
            .build();

        CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
        System.out.println("The campaign ARN is
"+campaignResponse.campaignArn());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateCampaign](#) in *AWS SDK for Java 2.x API Reference*.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.

import { CreateCampaignCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});
```

```
// Set the campaign's parameters.
export const createCampaignParam = {
  solutionVersionArn: 'SOLUTION_VERSION_ARN', /* required */
  name: 'NAME', /* required */
  minProvisionedTPS: 1 /* optional integer */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateCampaignCommand(createCampaignParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateCampaign](#) in *AWS SDK for JavaScript API Reference*.

## Create a dataset with Amazon Personalize using an AWS SDK

The following code examples show how to create a Amazon Personalize dataset.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createDataset(PersonalizeClient personalizeClient,
                                    String datasetName,
                                    String datasetGroupArn,
                                    String datasetType,
                                    String schemaArn) {
  try {
    CreateDatasetRequest request = CreateDatasetRequest.builder()
      .name(datasetName)
      .datasetGroupArn(datasetGroupArn)
      .datasetType(datasetType)
      .schemaArn(schemaArn)
      .build();

    String datasetArn = personalizeClient.createDataset(request)
      .datasetArn();
    System.out.println("Dataset " + datasetName + " created.");
    return datasetArn;
  } catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
  }
  return "";
}
```

- For API details, see [CreateDataset](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset's parameters.
export const createDatasetParam = {
  datasetGroupArn: 'DATASET_GROUP_ARN', /* required */
  datasetType: 'DATASET_TYPE', /* required */
  name: 'NAME', /* required */
  schemaArn: 'SCHEMA_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateDatasetCommand(createDatasetParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateDataset](#) in *AWS SDK for JavaScript API Reference*.

## Create a dataset export job with Amazon Personalize using an AWS SDK

The following code examples show how to create a Amazon Personalize dataset export job.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createDatasetExportJob(PersonalizeClient
personalizeClient,
                                              String jobName,
                                              String datasetArn,
                                              IngestionMode ingestionMode,
                                              String roleArn,
                                              String s3BucketPath,
                                              String kmsKeyArn) {
```

```
long waitInMilliseconds = 30 * 1000; // 30 seconds
String status = null;

try {

    S3DataConfig exportS3DataConfig =
S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
    DatasetExportJobOutput jobOutput =
DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig).build();

    CreateDatasetExportJobRequest createRequest =
CreateDatasetExportJobRequest.builder()
        .jobName(jobName)
        .datasetArn(datasetArn)
        .ingestionMode(ingestionMode)
        .jobOutput(jobOutput)
        .roleArn(roleArn)
        .build();

    String datasetExportJobArn =
personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

    DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
DescribeDatasetExportJobRequest.builder()
        .datasetExportJobArn(datasetExportJobArn)
        .build();

    long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    while (Instant.now().getEpochSecond() < maxTime) {

        DatasetExportJob datasetExportJob =
personalizeClient.describeDatasetExportJob(describeDatasetExportJobRequest)
            .datasetExportJob();

        status = datasetExportJob.status();
        System.out.println("Export job status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            return status;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}
```

- For API details, see [CreateDatasetExportJob](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetExportJobCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the export job parameters.
export const datasetExportJobParam = {
  datasetArn: 'DATASET_ARN', /* required */
  jobOutput: {
    s3DataDestination: {
      path: 'S3_DESTINATION_PATH' /* required */
      //kmsKeyArn: 'ARN' /* include if your bucket uses AWS KMS for encryption
    }
  },
  jobName: 'NAME',/* required */
  roleArn: 'ROLE_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateDatasetExportJobCommand(datasetExportJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateDatasetExportJob](#) in *AWS SDK for JavaScript API Reference*.

## Create a dataset group with Amazon Personalize using an AWS SDK

The following code examples show how to create a Amazon Personalize dataset group.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createDatasetGroup(PersonalizeClient personalizeClient,
String datasetGroupName) {

  try {
    CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
      .name(datasetGroupName)
      .build();
  return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
```

```
        } catch (PersonalizeException e) {
            System.out.println(e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

Create a domain dataset group.

```
public static String createDomainDatasetGroup(PersonalizeClient
personalizeClient,
                                              String datasetGroupName,
                                              String domain) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
        .name(datasetGroupName)
        .domain(domain)
        .build();
    return
    personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

- For API details, see [CreateDatasetGroup](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.

import { CreateDatasetGroupCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset group parameters.
export const createDatasetGroupParam = {
  name: 'NAME' /* required */
}

export const run = async (createDatasetGroupParam) => {
  try {
    const response = await personalizeClient.send(new
CreateDatasetGroupCommand(createDatasetGroupParam));
    console.log("Success", response);
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
}
```

```
    }  
};  
run(createDatasetGroupParam);
```

## Create a domain dataset group.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetGroupCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the domain dataset group parameters.
export const domainDatasetGroupParams = {
  name: 'NAME', /* required */
  domain: 'DOMAIN' /* required for a domain dsg, specify ECOMMERCE or
VIDEO_ON_DEMAND */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateDatasetGroupCommand(domainDatasetGroupParams));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};

run();
```

- For API details, see [CreateDatasetGroup](#) in *AWS SDK for JavaScript API Reference*.

## Create a dataset import job with Amazon Personalize using an AWS SDK

The following code examples show how to create a Amazon Personalize dataset import job.

Java

SDK for Java 2.x

## Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static String createPersonalizeDatasetImportJob(PersonalizeClient  
personalizeClient,  
                                                       String jobName,  
                                                       String datasetArn,  
                                                       String s3BucketPath,  
                                                       String roleArn) {  
  
    long waitInMilliseconds = 60 * 1000;  
    String status:
```

```
String datasetImportJobArn;

try {
    DataSource importDataSource = DataSource.builder()
        .dataLocation(s3BucketPath)
        .build();

    CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
    .datasetArn(datasetArn)
    .dataSource(importDataSource)
    .jobName(jobName)
    .roleArn(roleArn)
    .build();

    datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
    .datasetImportJobArn();
    DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
    .datasetImportJobArn(datasetImportJobArn)
    .build();

    long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    while (Instant.now().getEpochSecond() < maxTime) {

        DatasetImportJob datasetImportJob = personalizeClient
            .describeDatasetImportJob(describeDatasetImportJobRequest)
            .datasetImportJob();

        status = datasetImportJob.status();
        System.out.println("Dataset import job status: " + status);

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return datasetImportJobArn;
} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}
```

- For API details, see [CreateDatasetImportJob](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
```

```
import {CreateDatasetImportJobCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset import job parameters.
export const datasetImportJobParam = {
  datasetArn: 'DATASET_ARN', /* required */
  dataSource: { /* required */
    dataLocation: 'S3_PATH'
  },
  jobName: 'NAME',/* required */
  roleArn: 'ROLE_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateDatasetImportJobCommand(datasetImportJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateDatasetImportJob](#) in *AWS SDK for JavaScript API Reference*.

## Create a domain schema with Amazon Personalize using an AWS SDK

The following code examples show how to create a Amazon Personalize domain schema.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createDomainSchema(PersonalizeClient personalizeClient,
String schemaName, String domain, String filePath) {

    String schema = null;
    try {
        schema = new String(Files.readAllBytes(Paths.get(filePath)));
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }

    try {
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
            .name(schemaName)
            .domain(domain)
            .schema(schema)
```

```
        .build();

        String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

        System.out.println("Schema arn: " + schemaArn);

        return schemaArn;

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateSchema](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from 'fs';

let schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
  mySchema = 'TEST' // for unit tests.
}

// Set the domain schema parameters.
export const createDomainSchemaParam = {
  name: 'NAME', /* required */
  schema: mySchema, /* required */
  domain: 'DOMAIN' /* required for a domain dataset group, specify ECOMMERCE or
VIDEO_ON_DEMAND */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateSchemaCommand(createDomainSchemaParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
}
```

```
    }  
};  
run();
```

- For API details, see [CreateSchema](#) in *AWS SDK for JavaScript API Reference*.

## Create a filter with Amazon Personalize using an AWS SDK

The following code examples show how to create a Amazon Personalize filter.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createFilter(PersonalizeClient personalizeClient,  
                                  String filterName,  
                                  String datasetGroupArn,  
                                  String filterExpression) {  
    try {  
        CreateFilterRequest request = CreateFilterRequest.builder()  
            .name(filterName)  
            .datasetGroupArn(datasetGroupArn)  
            .filterExpression(filterExpression)  
            .build();  
  
        return personalizeClient.createFilter(request).filterArn();  
    }  
    catch(PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [CreateFilter](#) in *AWS SDK for Java 2.x API Reference*.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.  
import { CreateFilterCommand } from  
    "@aws-sdk/client-personalize";  
import { personalizeClient } from "./libs/personalizeClients.js";  
// Or, create the client here.  
// const personalizeClient = new PersonalizeClient({ region: "REGION"});
```

```
// Set the filter's parameters.
export const createFilterParam = {
  datasetGroupArn: 'DATASET_GROUP_ARN', /* required */
  name: 'NAME', /* required */
  filterExpression: 'FILTER_EXPRESSION' /*required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateFilterCommand(createFilterParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateFilter in AWS SDK for JavaScript API Reference](#).

## Create a recommender with Amazon Personalize using an AWS SDK

The following code examples show how to create a Amazon Personalize recommender.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createRecommender(PersonalizeClient personalizeClient,
                                         String name,
                                         String datasetGroupArn,
                                         String recipeArn) {

    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String recommenderStatus = "";

    try {
        CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
        .datasetGroupArn(datasetGroupArn)
        .name(name)
        .recipeArn(recipeArn)
        .build();

        CreateRecommenderResponse recommenderResponse =
personalizeClient.createRecommender(createRecommenderRequest);
        String recommenderArn = recommenderResponse.recommenderArn();
        System.out.println("The recommender ARN is " + recommenderArn);

        DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
        .recommenderArn(recommenderArn)
        .build();
    }
```

```
maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

while (Instant.now().getEpochSecond() < maxTime) {

    recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender().status();
    System.out.println("Recommender status: " + recommenderStatus);

    if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
        break;
    }
    try {
        Thread.sleep(waitInMilliseconds);
    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
    }
}
return recommenderArn;

} catch(PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [CreateRecommender](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateRecommenderCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the recommender's parameters.
export const createRecommenderParam = {
  name: 'NAME', /* required */
  recipeArn: 'RECIPE_ARN', /* required */
  datasetGroupArn: 'DATASET_GROUP_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateRecommenderCommand(createRecommenderParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
}
```

```
    }  
};  
run();
```

- For API details, see [CreateRecommender](#) in *AWS SDK for JavaScript API Reference*.

## Create a schema with Amazon Personalize using an AWS SDK

The following code examples show how to create a Amazon Personalize schema.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createSchema(PersonalizeClient personalizeClient, String  
schemaName, String filePath) {  
  
    String schema = null;  
    try {  
        schema = new String(Files.readAllBytes(Paths.get(filePath)));  
    } catch (IOException e) {  
        System.out.println(e.getMessage());  
    }  
  
    try {  
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()  
            .name(schemaName)  
            .schema(schema)  
            .build();  
  
        String schemaArn =  
personalizeClient.createSchema(createSchemaRequest).schemaArn();  
  
        System.out.println("Schema arn: " + schemaArn);  
  
        return schemaArn;  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [CreateSchema](#) in *AWS SDK for Java 2.x API Reference*.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from 'fs';

let schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
  mySchema = 'TEST' // For unit tests.
}
// Set the schema parameters.
export const createSchemaParam = {
  name: 'NAME', /* required */
  schema: mySchema /* required */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateSchemaCommand(createSchemaParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateSchema](#) in *AWS SDK for JavaScript API Reference*.

## Create a solution with Amazon Personalize using an AWS SDK

The following code examples show how to create a Amazon Personalize solution.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createPersonalizeSolution(PersonalizeClient
personalizeClient,
                                                String datasetGroupArn,
                                                String solutionName,
                                                String recipeArn) {

  try {
    CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
      .name(solutionName)
```

```
        .datasetGroupArn(datasetGroupArn)
        .recipeArn(recipeArn)
        .build();

    CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
    return solutionResponse.solutionArn();

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [CreateSolution](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateSolutionCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the solution parameters.
export const createSolutionParam = {
  datasetGroupArn: 'DATASET_GROUP_ARN', /* required */
  recipeArn: 'RECIPE_ARN', /* required */
  name: 'NAME' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateSolutionCommand(createSolutionParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateSolution](#) in *AWS SDK for JavaScript API Reference*.

## Create a solution version with Amazon Personalize using an AWS SDK

The following code examples show how to create a Amazon Personalize solution.

Java

**SDK for Java 2.x**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String solutionStatus = "";
    String solutionVersionStatus = "";
    String solutionVersionArn = "";

    try {
        DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
        .solutionArn(solutionArn)
        .build();

        maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        // Wait until solution is active.
        while (Instant.now().getEpochSecond() < maxTime) {

            solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
            System.out.println("Solution status: " + solutionStatus);

            if (solutionStatus.equals("ACTIVE") ||
solutionStatus.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }

        if (solutionStatus.equals("ACTIVE")) {

            CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
            .solutionArn(solutionArn)
            .build();

            CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient.createSolutionVersion(createSolutionVersionRequest);
            solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

            System.out.println("Solution version ARN: " + solutionVersionArn);

            DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
            .solutionVersionArn(solutionVersionArn)
            .build();

            while (Instant.now().getEpochSecond() < maxTime) {
```

```
solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest).solutionVersion().status
System.out.println("Solution version status: " +
solutionVersionStatus);

        if (solutionVersionStatus.equals("ACTIVE") || solutionVersionStatus.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return solutionVersionArn;
}
} catch(PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [CreateSolutionVersion in AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateSolutionVersionCommand } from
    "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the solution version parameters.
export const solutionVersionParam = {
    solutionArn: 'SOLUTION_ARN' /* required */
}

export const run = async () => {
    try {
        const response = await personalizeClient.send(new
CreateSolutionVersionCommand(solutionVersionParam));
        console.log("Success", response);
        return response; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For API details, see [CreateSolutionVersion in AWS SDK for JavaScript API Reference](#).

## Create an event tracker with Amazon Personalize using an AWS SDK

The following code examples show how to create a Amazon Personalize event tracker.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createEventTracker(PersonalizeClient personalizeClient,
String eventTrackerName, String datasetGroupArn) {

    String eventTrackerId = "";
    String eventTrackerArn;
    long maxTime = 3 * 60 * 60; // 3 hours
    long waitInMilliseconds = 20 * 1000; // 20 seconds
    String status;

    try {

        CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
            .name(eventTrackerName)
            .datasetGroupArn(datasetGroupArn)
            .build();

        CreateEventTrackerResponse createEventTrackerResponse =
personalizeClient.createEventTracker(createEventTrackerRequest);

        eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
        eventTrackerId = createEventTrackerResponse.trackingId();
        System.out.println("Event tracker ARN: " + eventTrackerArn);
        System.out.println("Event tracker ID: " + eventTrackerId);

        maxTime = Instant.now().getEpochSecond() + maxTime;

        DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
            .eventTrackerArn(eventTrackerArn)
            .build();

        while (Instant.now().getEpochSecond() < maxTime) {

            status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
            System.out.println("EventTracker status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    return eventTrackerId;
}
```

```
        }
        catch (PersonalizeException e){
            System.out.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return eventTrackerId;
    }
```

- For API details, see [CreateEventTracker](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateEventTrackerCommand } from
    "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the event tracker's parameters.
export const createEventTrackerParam = {
    datasetGroupArn: 'DATASET_GROUP_ARN', /* required */
    name: 'NAME', /* required */
}

export const run = async () => {
    try {
        const response = await personalizeClient.send(new
CreateEventTrackerCommand(createEventTrackerParam));
        console.log("Success", response);
        return response; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For API details, see [CreateEventTracker](#) in *AWS SDK for JavaScript API Reference*.

## Delete a campaign in Amazon Personalize using an AWS SDK

The following code example shows how to delete a campaign in Amazon Personalize.

### Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,  
String campaignArn ) {  
  
    try {  
        DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()  
            .campaignArn(campaignArn)  
            .build();  
  
        personalizeClient.deleteCampaign(campaignRequest);  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteCampaign](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a solution in Amazon Personalize using an AWS SDK

The following code example shows how to delete a solution in Amazon Personalize.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteGivenSolution(PersonalizeClient personalizeClient,  
String solutionArn ) {  
  
    try {  
        DeleteSolutionRequest solutionRequest = DeleteSolutionRequest.builder()  
            .solutionArn(solutionArn)  
            .build();  
  
        personalizeClient.deleteSolution(solutionRequest);  
        System.out.println("Done");  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteSolution](#) in *AWS SDK for Java 2.x API Reference*.

## Delete an event tracker in Amazon Personalize using an AWS SDK

The following code example shows how to delete an event tracker in Amazon Personalize.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteEventTracker(PersonalizeClient personalizeClient,
String eventTrackerArn) {
    try {
        DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
    .eventTrackerArn(eventTrackerArn)
    .build();

    int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCode()

        System.out.println("Status code:" + status);

    }
    catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteEventTracker](#) in *AWS SDK for Java 2.x API Reference*.

## Describe a campaign in Amazon Personalize using an AWS SDK

The following code example shows how to describe a campaign in Amazon Personalize.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeSpecificCampaign(PersonalizeClient
personalizeClient, String campaignArn) {

    try {
        DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
    .campaignArn(campaignArn)
    .build();

        DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
        Campaign myCampaign = campaignResponse.campaign();
        System.out.println("The Campaign name is "+myCampaign.name());
        System.out.println("The Campaign status is "+myCampaign.status());
    }
}
```

```
        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
    }
}
```

- For API details, see [DescribeCampaign](#) in *AWS SDK for Java 2.x API Reference*.

## Describe a recipe in Amazon Personalize using an AWS SDK

The following code example shows how to describe a recipe in Amazon Personalize.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeSpecificRecipe(PersonalizeClient personalizeClient,
String recipeArn) {

    try{
        DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()
            .recipeArn(recipeArn)
            .build();

        DescribeRecipeResponse recipeResponse =
personalizeClient.describeRecipe(recipeRequest);
        System.out.println("The recipe name is
"+recipeResponse.recipe().name());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeRecipe](#) in *AWS SDK for Java 2.x API Reference*.

## Describe a solution in Amazon Personalize using an AWS SDK

The following code example shows how to describe a solution in Amazon Personalize.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeSpecificSolution(PersonalizeClient personalizeClient,
String solutionArn) {
```

```
try {
    DescribeSolutionRequest solutionRequest =
DescribeSolutionRequest.builder()
    .solutionArn(solutionArn)
    .build();

    DescribeSolutionResponse response =
personalizeClient.describeSolution(solutionRequest);
    System.out.println("The Solution name is "+response.solution().name());

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [DescribeSolution](#) in *AWS SDK for Java 2.x API Reference*.

## Incrementally import an item into Amazon Personalize Events using an AWS SDK

The following code example shows how to import real-time interaction event data into Amazon Personalize Events.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { PutItemsCommand } from
  "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region:
  "REGION"});

// Set the put items parameters. For string properties and values, use the \
character to escape quotes.
var putItemsParam = {
  datasetArn: 'DATASET_ARN', /* required */
  items: [ /* required */
    {
      'itemId': 'ITEM_ID', /* required */
      'properties': "{\"PROPERTY1_NAME\": \"PROPERTY1_VALUE\", \"PROPERTY2_NAME\
\": \"PROPERTY2_VALUE\", \"PROPERTY3_NAME\": \"PROPERTY3_VALUE\"}" /* optional */
    }
  ];
};

export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(new
PutItemsCommand(putItemsParam));
    console.log("Success!", response);
    return response; // For unit tests.
  }
};
```

```
        } catch (err) {
            console.log("Error", err);
        }
    };
    run();
}
```

- For API details, see [PutItems](#) in *AWS SDK for JavaScript API Reference*.

## List campaigns in Amazon Personalize using an AWS SDK

The following code example shows how to list campaigns in Amazon Personalize.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllCampaigns(PersonalizeClient personalizeClient, String
solutionArn) {

    try{
        ListCampaignsRequest campaignsRequest = ListCampaignsRequest.builder()
            .maxResults(10)
            .solutionArn(solutionArn)
            .build();

        ListCampaignsResponse response =
personalizeClient.listCampaigns(campaignsRequest);
        List<CampaignSummary> campaigns = response.campaigns();
        for (CampaignSummary campaign: campaigns) {
            System.out.println("Campaign name is : "+campaign.name());
            System.out.println("Campaign ARN is : "+campaign.campaignArn());
        }

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListCampaigns](#) in *AWS SDK for Java 2.x API Reference*.

## List dataset groups in Amazon Personalize using an AWS SDK

The following code example shows how to list dataset groups in Amazon Personalize.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listDSGroups( PersonalizeClient personalizeClient ) {

    try {
        ListDatasetGroupsRequest groupsRequest =
ListDatasetGroupsRequest.builder()
            .maxResults(15)
            .build();

        ListDatasetGroupsResponse groupsResponse =
personalizeClient.listDatasetGroups(groupsRequest);
        List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();
        for (DatasetGroupSummary group: groups) {
            System.out.println("The DataSet name is : "+group.name());
            System.out.println("The DataSet ARN is :
"+group.datasetGroupArn());
        }
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListDatasetGroups](#) in *AWS SDK for Java 2.x API Reference*.

## List recipes in Amazon Personalize using an AWS SDK

The following code example shows how to list recipes in Amazon Personalize.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllRecipes(PersonalizeClient personalizeClient) {

    try {
        ListRecipesRequest recipesRequest = ListRecipesRequest.builder()
            .maxResults(15)
            .build();

        ListRecipesResponse response =
personalizeClient.listRecipes(recipesRequest);
        List<RecipeSummary> recipes = response.recipes();
        for (RecipeSummary recipe: recipes) {
            System.out.println("The recipe ARN is: "+recipe.recipeArn());
            System.out.println("The recipe name is: "+recipe.name());
        }
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListRecipes](#) in *AWS SDK for Java 2.x API Reference*.

## List solutions in Amazon Personalize using an AWS SDK

The following code example shows how to list solutions in Amazon Personalize.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllSolutions(PersonalizeClient personalizeClient, String datasetGroupArn) {  
  
    try {  
        ListSolutionsRequest solutionsRequest = ListSolutionsRequest.builder()  
            .maxResults(10)  
            .datasetGroupArn(datasetGroupArn)  
            .build();  
  
        ListSolutionsResponse response =  
personalizeClient.listSolutions(solutionsRequest);  
        List<SolutionSummary> solutions = response.solutions();  
        for (SolutionSummary solution: solutions) {  
            System.out.println("The solution ARN is: "+solution.solutionArn());  
            System.out.println("The solution name is: "+solution.name());  
        }  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListSolutions](#) in *AWS SDK for Java 2.x API Reference*.

## Update a campaign in Amazon Personalize using an AWS SDK

The following code example shows how to update a campaign Amazon Personalize.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String updateCampaign(PersonalizeClient personalizeClient,  
String campaignArn,  
String solutionVersionArn,  
Integer minProvisionedTPS) {
```

```
try {
    // build the updateCampaignRequest
    UpdateCampaignRequest updateCampaignRequest =
UpdateCampaignRequest.builder()
    .campaignArn(campaignArn)
    .solutionVersionArn(solutionVersionArn)
    .minProvisionedTPS(minProvisionedTPS)
    .build();

    // update the campaign
    personalizeClient.updateCampaign(updateCampaignRequest);

    DescribeCampaignRequest campaignRequest =
DescribeCampaignRequest.builder()
    .campaignArn(campaignArn)
    .build();

    DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
    Campaign updatedCampaign = campaignResponse.campaign();

    System.out.println("The Campaign status is " +
updatedCampaign.status());
    return updatedCampaign.status();

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [UpdateCampaign](#) in *AWS SDK for Java 2.x API Reference*.

## Code examples for Amazon Personalize Events using AWS SDKs

The following code examples show how to use Amazon Personalize Events with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon Personalize Events using AWS SDKs \(p. 1363\)](#)
  - [Import real-time interaction event data into Amazon Personalize Events using an AWS SDK \(p. 1364\)](#)
  - [Incrementally import users into Amazon Personalize Events Events using an AWS SDK \(p. 1365\)](#)

## Actions for Amazon Personalize Events using AWS SDKs

The following code examples show how to use Amazon Personalize Events with AWS SDKs. Each example calls an individual service function.

### Examples

- Import real-time interaction event data into Amazon Personalize Events using an AWS SDK (p. 1364)
- Incrementally import users into Amazon Personalize Events Events using an AWS SDK (p. 1365)

## Import real-time interaction event data into Amazon Personalize Events using an AWS SDK

The following code examples show how to import real-time interaction event data into Amazon Personalize Events.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static int putItems(PersonalizeEventsClient personalizeEventsClient,
                           String datasetArn,
                           String item1Id,
                           String item1PropertyName,
                           String item1PropertyValue,
                           String item2Id,
                           String item2PropertyName,
                           String item2PropertyValue) {

    int responseCode = 0;
    ArrayList<Item> items = new ArrayList<>();

    try {
        Item item1 = Item.builder()
            .itemId(item1Id)
            .properties(String.format("{\"%1$s\": \"%2$s\"}",
                                      item1PropertyName, item1PropertyValue))
            .build();

        items.add(item1);

        Item item2 = Item.builder()
            .itemId(item2Id)
            .properties(String.format("{\"%1$s\": \"%2$s\"}",
                                      item2PropertyName, item2PropertyValue))
            .build();

        items.add(item2);

        PutItemsRequest putItemsRequest = PutItemsRequest.builder()
            .datasetArn(datasetArn)
            .items(items)
            .build();

        responseCode =
personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;
    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
}
```

```
}
```

- For API details, see [PutEvents](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { PutEventsCommand } from
  "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region:
  "REGION"});

// Convert your UNIX timestamp to a Date.
const sentAtDate = new Date(1613443801 * 1000) // 1613443801 is a testing value.
Replace it with your sentAt timestamp in UNIX format.

// Set put events parameters.
var putEventsParam = {
  eventList: [ /* required */
    {
      eventType: 'EVENT_TYPE', /* required */
      sentAt: sentAtDate, /* required, must be a Date with js */
      eventId: 'EVENT_ID', /* optional */
      itemId: 'ITEM_ID' /* optional */
    }
  ],
  sessionId: 'SESSION_ID', /* required */
  trackingId: 'TRACKING_ID', /* required */
  userId: 'USER_ID' /* required */
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(new
PutEventsCommand(putEventsParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [PutEvents](#) in *AWS SDK for JavaScript API Reference*.

## Incrementally import users into Amazon Personalize Events Events using an AWS SDK

The following code examples show how to incrementally import a user into Amazon Personalize Events Events.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static int putUsers(PersonalizeEventsClient personalizeEventsClient,
    String datasetArn,
    String user1Id,
    String user1PropertyName,
    String user1PropertyValue,
    String user2Id,
    String user2PropertyName,
    String user2PropertyValue) {

    int responseCode = 0;
    ArrayList<User> users = new ArrayList<>();

    try {
        User user1 = User.builder()
            .userId(user1Id)
            .properties(String.format("{\"%1$s\": \"%2$s\"}", user1PropertyName, user1PropertyValue))
            .build();

        users.add(user1);

        User user2 = User.builder()
            .userId(user2Id)
            .properties(String.format("{\"%1$s\": \"%2$s\"}", user2PropertyName, user2PropertyValue))
            .build();

        users.add(user2);

        PutUsersRequest putUsersRequest = PutUsersRequest.builder()
            .datasetArn(datasetArn)
            .users(users)
            .build();

        responseCode =
personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;
    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
}
```

- For API details, see [PutUsers](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { PutUsersCommand } from
  "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region:
  "REGION"});

// Set the put users parameters. For string properties and values, use the \
character to escape quotes.
var putUsersParam = {
  datasetArn: "DATASET_ARN",
  users: [
    {
      'userId': 'USER_ID',
      'properties': "{\"PROPERTY1_NAME\": \"PROPERTY1_VALUE\"}"
    }
  ]
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(new
PutUsersCommand(putUsersParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [PutUsers](#) in *AWS SDK for JavaScript API Reference*.

## Code examples for Amazon Personalize Runtime using AWS SDKs

The following code examples show how to use Amazon Personalize Runtime with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon Personalize Runtime using AWS SDKs \(p. 1367\)](#)
  - [Get ranked recommendations from a campaign created in a custom dataset group with Amazon Personalize Runtime using an AWS SDK \(p. 1368\)](#)
  - [Get recommendations from a campaign created in a custom dataset group with Amazon Personalize Runtime using an AWS SDK \(p. 1369\)](#)

## Actions for Amazon Personalize Runtime using AWS SDKs

The following code examples show how to use Amazon Personalize Runtime with AWS SDKs. Each example calls an individual service function.

### Examples

- Get ranked recommendations from a campaign created in a custom dataset group with Amazon Personalize Runtime using an AWS SDK (p. 1368)
- Get recommendations from a campaign created in a custom dataset group with Amazon Personalize Runtime using an AWS SDK (p. 1369)

## Get ranked recommendations from a campaign created in a custom dataset group with Amazon Personalize Runtime using an AWS SDK

The following code examples show how to get Amazon Personalize Runtime ranked recommendations.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
                                                 String campaignArn,
                                                 String userId,
                                                 ArrayList<String> items) {

    try {
        GetPersonalizedRankingRequest rankingRecommendationsRequest =
GetPersonalizedRankingRequest.builder()
        .campaignArn(campaignArn)
        .userId(userId)
        .inputList(items)
        .build();

        GetPersonalizedRankingResponse recommendationsResponse =
personalizeRuntimeClient.getPersonalizedRanking(rankingRecommendationsRequest);
        List<PredictedItem> rankedItems =
recommendationsResponse.personalizedRanking();
        int rank = 1;
        for (PredictedItem item : rankedItems) {
            System.out.println("Item ranked at position " + rank + " details");
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());

System.out.println("-----");
            rank++;
        }
        return rankedItems;
    } catch (PersonalizeRuntimeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
```

- For API details, see [GetPersonalizedRanking](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { GetPersonalizedRankingCommand } from
  "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
  "REGION"});

// Set the ranking request parameters.
export const getPersonalizedRankingParam = {
  campaignArn: "CAMPAIGN_ARN", /* required */
  userId: 'USER_ID', /* required */
  inputList: ["ITEM_ID_1", "ITEM_ID_2", "ITEM_ID_3", "ITEM_ID_4"]
}

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
      GetPersonalizedRankingCommand(getPersonalizedRankingParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [GetPersonalizedRanking](#) in *AWS SDK for JavaScript API Reference*.

## Get recommendations from a campaign created in a custom dataset group with Amazon Personalize Runtime using an AWS SDK

The following code examples show how to get Amazon Personalize Runtime Runtime recommendations.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get a list of recommended items.

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
  String campaignArn, String userId){
```

```

try {
    GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
    .campaignArn(campaignArn)
    .numResults(20)
    .userId(userId)
    .build();

    GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient.getRecommendations(recommendationsRequest);
    List<PredictedItem> items = recommendationsResponse.itemList();
    for (PredictedItem item: items) {
        System.out.println("Item Id is : "+item.itemId());
        System.out.println("Item score is : "+item.score());
    }

} catch (AwsServiceException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

```

Get a list of recommended items from a recommender created in a domain dataset group.

```

public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String recommenderArn, String userId){

    try {
        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
    .recommenderArn(recommenderArn)
    .numResults(20)
    .userId(userId)
    .build();

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient.getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();

        for (PredictedItem item: items) {
            System.out.println("Item Id is : "+item.itemId());
            System.out.println("Item score is : "+item.score());
        }

    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

Use a filter when requesting recommendations.

```

public static void getFilteredRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
                                    String campaignArn,
                                    String userId,
                                    String filterArn,
                                    String parameter1Name,
                                    String parameter1Value1,
                                    String parameter1Value2,
                                    String parameter2Name,
                                    String parameter2Value){

```

```
try {

    Map<String, String> filterValues = new HashMap<>();

    filterValues.put(parameter1Name, String.format("\\"%1$s\\",\\"%2$s\\",
        parameter1Value1, parameter1Value2));
    filterValues.put(parameter2Name, String.format("\\"%1$s\\",
        parameter2Value));

    GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
    .campaignArn(campaignArn)
    .numResults(20)
    .userId(userId)
    .filterArn(filterArn)
    .filterValues(filterValues)
    .build();

    GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient.getRecommendations(recommendationsRequest);
    List<PredictedItem> items = recommendationsResponse.itemList();

    for (PredictedItem item: items) {
        System.out.println("Item Id is : "+item.itemId());
        System.out.println("Item score is : "+item.score());
    }
} catch (PersonalizeRuntimeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [GetRecommendations](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from
    "@aws-sdk/client-personalize-runtime";

import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
    "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
    campaignArn: 'CAMPAIGN_ARN', /* required */
    userId: 'USER_ID', /* required */
    numResults: 15 /* optional */
}

export const run = async () => {
    try {
```

```
    const response = await personalizeRuntimeClient.send(new
GetRecommendationsCommand(getRecommendationsParam));
    console.log("Success!", response);
    return response; // For unit tests.
} catch (err) {
    console.log("Error", err);
}
};

run();
```

Get recommendation with a filter (custom dataset group).

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from
    "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
    "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
    recommenderArn: 'RECOMMENDER_ARN', /* required */
    userId: 'USER_ID', /* required */
    numResults: 15 /* optional */
}

export const run = async () => {
    try {
        const response = await personalizeRuntimeClient.send(new
GetRecommendationsCommand(getRecommendationsParam));
        console.log("Success!", response);
        return response; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

Get filtered recommendations from a recommender created in a domain dataset group.

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from
    "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
    "REGION"});

// Set recommendation request parameters.
export const getRecommendationsParam = {
    campaignArn: 'CAMPAIGN_ARN', /* required */
    userId: 'USER_ID', /* required */
    numResults: 15, /* optional */
    filterArn: 'FILTER_ARN', /* required to filter recommendations */
    filterValues: {
        "PROPERTY": "\"VALUE\""
            /* Only required if your filter has a placeholder
            parameter */
    }
}
```

```
export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
GetRecommendationsCommand(getRecommendationsParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [GetRecommendations in AWS SDK for JavaScript API Reference](#).

## Code examples for Amazon Pinpoint using AWS SDKs

The following code examples show how to use Amazon Pinpoint with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon Pinpoint using AWS SDKs \(p. 1373\)](#)
  - [Create an Amazon Pinpoint campaign \(p. 1374\)](#)
  - [Create an Amazon Pinpoint segment \(p. 1376\)](#)
  - [Create an Amazon Pinpoint application \(p. 1378\)](#)
  - [Delete an Amazon Pinpoint application \(p. 1379\)](#)
  - [Delete an Amazon Pinpoint endpoint \(p. 1380\)](#)
  - [Export an Amazon Pinpoint endpoint \(p. 1381\)](#)
  - [Display information about an existing Amazon Pinpoint endpoint \(p. 1384\)](#)
  - [Import an Amazon Pinpoint segment \(p. 1385\)](#)
  - [List Amazon Pinpoint endpoints that are associated with a specific user ID \(p. 1386\)](#)
  - [List Amazon Pinpoint segments in an application \(p. 1387\)](#)
  - [Send email and text messages with Amazon Pinpoint using an AWS SDK \(p. 1388\)](#)
  - [Send templated email and text messages with Amazon Pinpoint using an AWS SDK \(p. 1401\)](#)
  - [Update an Amazon Pinpoint endpoint \(p. 1403\)](#)
  - [Update an Amazon Pinpoint channel \(p. 1405\)](#)

## Actions for Amazon Pinpoint using AWS SDKs

The following code examples show how to use Amazon Pinpoint with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create an Amazon Pinpoint campaign \(p. 1374\)](#)
- [Create an Amazon Pinpoint segment \(p. 1376\)](#)
- [Create an Amazon Pinpoint application \(p. 1378\)](#)

- [Delete an Amazon Pinpoint application \(p. 1379\)](#)
- [Delete an Amazon Pinpoint endpoint \(p. 1380\)](#)
- [Export an Amazon Pinpoint endpoint \(p. 1381\)](#)
- [Display information about an existing Amazon Pinpoint endpoint \(p. 1384\)](#)
- [Import an Amazon Pinpoint segment \(p. 1385\)](#)
- [List Amazon Pinpoint endpoints that are associated with a specific user ID \(p. 1386\)](#)
- [List Amazon Pinpoint segments in an application \(p. 1387\)](#)
- [Send email and text messages with Amazon Pinpoint using an AWS SDK \(p. 1388\)](#)
- [Send templated email and text messages with Amazon Pinpoint using an AWS SDK \(p. 1401\)](#)
- [Update an Amazon Pinpoint endpoint \(p. 1403\)](#)
- [Update an Amazon Pinpoint channel \(p. 1405\)](#)

## Create an Amazon Pinpoint campaign

The following code examples show how to create a campaign.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a campaign.

```
public static void createPinCampaign(PinpointClient pinpoint, String appId,
String segmentId) {
    CampaignResponse result = createCampaign(pinpoint, appId, segmentId);
    System.out.println("Campaign " + result.name() + " created.");
    System.out.println(result.description());
}

public static CampaignResponse createCampaign(PinpointClient client, String
appID, String segmentID) {

    try {
        Schedule schedule = Schedule.builder()
            .startTime("IMMEDIATE")
            .build();

        Message defaultMessage = Message.builder()
            .action(Action.OPEN_APP)
            .body("My message body.")
            .title("My message title.")
            .build();

        MessageConfiguration messageConfiguration =
MessageConfiguration.builder()
            .defaultMessage(defaultMessage)
            .build();

        WriteCampaignRequest request = WriteCampaignRequest.builder()
            .description("My description")
            .schedule(schedule)
            .name("MyCampaign")
            .segmentId(segmentID)
    }
}
```

```
        .messageConfiguration(messageConfiguration)
        .build();

        CreateCampaignResponse result =
client.createCampaign(CreateCampaignRequest.builder()
                    .applicationId(appID)
                    .writeCampaignRequest(request).build()
            );
        System.out.println("Campaign ID: " + result.campaignResponse().id());
        return result.campaignResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- For API details, see [CreateCampaign](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createPinCampaign(appId: String, segmentIdVal: String) {

    val schedule0b = Schedule {
        startTime = "IMMEDIATE"
    }

    val defaultMessage0b = Message {
        action = Action.OpenApp
        body = "My message body"
        title = "My message title"
    }

    val messageConfiguration0b = MessageConfiguration {
        defaultMessage = defaultMessage0b
    }

    val writeCampaign = WriteCampaignRequest {
        description = "My description"
        schedule = schedule0b
        name = "MyCampaign"
        segmentId = segmentIdVal
        messageConfiguration = messageConfiguration0b
    }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse = pinpoint.createCampaign(
            CreateCampaignRequest {

```

```
        applicationId = appId
        writeCampaignRequest = writeCampaign
    }
}
println("Campaign ID is ${result.campaignResponse?.id}")
}
```

- For API details, see [CreateCampaign](#) in *AWS SDK for Kotlin API reference*.

## Create an Amazon Pinpoint segment

The following code examples show how to create a segment.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static SegmentResponse createSegment(PinpointClient client, String
appId) {

    try {
        Map<String, AttributeDimension> segmentAttributes = new HashMap<>();
        segmentAttributes.put("Team", AttributeDimension.builder()
            .attributeType(AttributeType.INCLUSIVE)
            .values("Lakers")
            .build());

        RecencyDimension recencyDimension = RecencyDimension.builder()
            .duration("DAY_30")
            .recencyType("ACTIVE")
            .build();

        SegmentBehaviors segmentBehaviors = SegmentBehaviors.builder()
            .recency(recencyDimension)
            .build();

        SegmentDemographics segmentDemographics = SegmentDemographics
            .builder()
            .build();

        SegmentLocation segmentLocation = SegmentLocation
            .builder()
            .build();

        SegmentDimensions dimensions = SegmentDimensions
            .builder()
            .attributes(segmentAttributes)
            .behavior(segmentBehaviors)
            .demographic(segmentDemographics)
            .location(segmentLocation)
            .build();

        WriteSegmentRequest writeSegmentRequest = WriteSegmentRequest.builder()
            .name("MySegment")
            .dimensions(dimensions)
    }
}
```

```
.build();

CreateSegmentRequest createSegmentRequest =
CreateSegmentRequest.builder()
    .applicationId(appId)
    .writeSegmentRequest(writeSegmentRequest)
    .build();

CreateSegmentResponse createSegmentResult =
client.createSegment(createSegmentRequest);
System.out.println("Segment ID: " +
createSegmentResult.segmentResponse().id());
System.out.println("Done");
return createSegmentResult.segmentResponse();

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}
```

- For API details, see [CreateSegment](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createPinpointSegment(applicationIdVal: String?): String? {

    val segmentAttributes = mutableMapOf<String, AttributeDimension>()
    val myList = mutableListOf<String>()
    myList.add("Lakers")

    val atts = AttributeDimension {
        attributeType = AttributeType.Inclusive
        values = myList
    }

    segmentAttributes["Team"] = atts
    val recencyDimension = RecencyDimension {
        duration = Duration.fromValue("DAY_30")
        recencyType = RecencyType.fromValue("ACTIVE")
    }

    val segmentBehaviors = SegmentBehaviors {
        recency = recencyDimension
    }

    val segmentLocation = SegmentLocation {}
    val dimensionsOb = SegmentDimensions {
        attributes = segmentAttributes
        behavior = segmentBehaviors
    }
}
```

```
    demographic = SegmentDemographics {}
    location = segmentLocation
}

val writeSegmentRequest0b = WriteSegmentRequest {
    name = "MySegment101"
    dimensions = dimensions0b
}

PinpointClient { region = "us-west-2" }.use { pinpoint ->
    val createSegmentResult: CreateSegmentResponse = pinpoint.createSegment(
        CreateSegmentRequest {
            applicationId = applicationIdVal
            writeSegmentRequest = writeSegmentRequest0b
        }
    )
    println("Segment ID is ${createSegmentResult.segmentResponse?.id}")
    return createSegmentResult.segmentResponse?.id
}
}
```

- For API details, see [CreateSegment](#) in *AWS SDK for Kotlin API reference*.

## Create an Amazon Pinpoint application

The following code examples show how to create an application.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createApplication(PinpointClient pinpoint, String appName)
{
    try {
        CreateApplicationRequest appRequest =
CreateApplicationRequest.builder()
            .name(appName)
            .build();

        CreateAppRequest request = CreateAppRequest.builder()
            .createApplicationRequest(appRequest)
            .build();

        CreateAppResponse result = pinpoint.createApp(request);
        return result.applicationResponse().id();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateApp](#) in *AWS SDK for Java 2.x API Reference*.

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createApplication(applicationName: String?): String? {  
  
    val createApplicationRequest0b = CreateApplicationRequest {  
        name = applicationName  
    }  
  
    PinpointClient { region = "us-west-2" }.use { pinpoint ->  
        val result = pinpoint.createApp(  
            CreateAppRequest {  
                createApplicationRequest = createApplicationRequest0b  
            }  
        )  
        return result.applicationResponse?.id  
    }  
}
```

- For API details, see [CreateApp](#) in *AWS SDK for Kotlin API reference*.

## Delete an Amazon Pinpoint application

The following code examples show how to delete an application.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an application.

```
public static void deletePinApp(PinpointClient pinpoint, String appId ) {  
  
    try {  
        DeleteAppRequest appRequest = DeleteAppRequest.builder()  
            .applicationId(appId)  
            .build();  
  
        DeleteAppResponse result = pinpoint.deleteApp(appRequest);  
        String appName = result.applicationResponse().name();  
        System.out.println("Application " + appName + " has been deleted.");  
  
    } catch (PinpointException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteApp](#) in *AWS SDK for Java 2.x API Reference*.

Kotlin

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deletePinApp(appId: String?) {  
  
    PinpointClient { region = "us-west-2" }.use { pinpoint ->  
        val result = pinpoint.deleteApp(  
            DeleteAppRequest {  
                applicationId = appId  
            }  
        )  
        val appName = result.applicationResponse?.name  
        println("Application $appName has been deleted.")  
    }  
}
```

- For API details, see [DeleteApp](#) in *AWS SDK for Kotlin API reference*.

## Delete an Amazon Pinpoint endpoint

The following code examples show how to delete an endpoint.

Java

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an endpoint.

```
public static void deletePinEncpoint(PinpointClient pinpoint, String appId,  
String endpointId ) {  
  
    try {  
        DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()  
            .applicationId(appId)  
            .endpointId(endpointId)  
            .build();  
  
        DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);  
        String id = result.endpointResponse().id();  
        System.out.println("The deleted endpoint id " + id);  
  
    } catch (PinpointException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

- For API details, see [DeleteEndpoint](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deletePinEndpoint(appIdVal: String?, endpointIdVal: String?) {

    val deleteEndpointRequest = DeleteEndpointRequest {
        applicationId = appIdVal
        endpointId = endpointIdVal
    }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.deleteEndpoint(deleteEndpointRequest)
        val id = result.endpointResponse?.id
        println("The deleted endpoint is $id")
    }
}
```

- For API details, see [DeleteEndpoint](#) in *AWS SDK for Kotlin API reference*.

## Export an Amazon Pinpoint endpoint

The following code example shows how to export an endpoint.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Export an endpoint.

```
public static void exportAllEndpoints(PinpointClient pinpoint,
                                      S3Client s3Client,
                                      String applicationId,
                                      String s3BucketName,
                                      String path,
                                      String iamExportRoleArn) {

    try {
```

```

        List<String> objectKeys = exportEndpointsToS3(pinpoint, s3Client,
s3BucketName, iamExportRoleArn, applicationId);
        List<String> endpointFileKeys = objectKeys.stream().filter(o ->
o.endsWith(".gz")).collect(Collectors.toList());
        downloadFromS3(s3Client, path, s3BucketName, endpointFileKeys);

    } catch ( PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<String> exportEndpointsToS3(PinpointClient pinpoint,
S3Client s3Client, String s3BucketName, String iamExportRoleArn, String
applicationId) {

    SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH_mm:ss.SSS_z");
    String endpointsKeyPrefix = "exports/" + applicationId + "_" +
dateFormat.format(new Date());
    String s3UrlPrefix = "s3://" + s3BucketName + "/" + endpointsKeyPrefix +
"/";
    List<String> objectKeys = new ArrayList<>();
    String key;

    try {
        // Defines the export job that Amazon Pinpoint runs.
        ExportJobRequest jobRequest = ExportJobRequest.builder()
            .roleArn(iamExportRoleArn)
            .s3UrlPrefix(s3UrlPrefix)
            .build();

        CreateExportJobRequest exportJobRequest =
CreateExportJobRequest.builder()
            .applicationId(applicationId)
            .exportJobRequest(jobRequest)
            .build();

        System.out.format("Exporting endpoints from Amazon Pinpoint application
%s to Amazon S3 " +
"bucket %s . . .\n", applicationId, s3BucketName);

        CreateExportJobResponse exportResult =
pinpoint.createExportJob(exportJobRequest);
        String jobId = exportResult.exportJobResponse().id();
        System.out.println(jobId);
        printExportJobStatus(pinpoint, applicationId, jobId);

        ListObjectsV2Request v2Request = ListObjectsV2Request.builder()
            .bucket(s3BucketName)
            .prefix(endpointsKeyPrefix)
            .build();

        // Create a list of object keys.
        ListObjectsV2Response v2Response = s3Client.listObjectsV2(v2Request);
        List<S3Object> objects = v2Response.contents();
        for (S3Object object: objects) {
            key = object.key();
            objectKeys.add(key);
        }

        return objectKeys;
    } catch ( PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

```

        }

    }

    private static void printExportJobStatus(PinpointClient pinpointClient,
                                            String applicationId,
                                            String jobId) {

        GetExportJobResponse getExportJobResult;
        String status;

        try {
            // Checks the job status until the job completes or fails.
            GetExportJobRequest exportJobRequest = GetExportJobRequest.builder()
                .jobId(jobId)
                .applicationId(applicationId)
                .build();

            do {
                getExportJobResult = pinpointClient.getExportJob(exportJobRequest);
                status =
getExportJobResult.exportJobResponse().jobStatus().toString().toUpperCase();
                System.out.format("Export job %s . . .\n", status);
                TimeUnit.SECONDS.sleep(3);

            } while (!status.equals("COMPLETED") && !status.equals("FAILED"));

            if (status.equals("COMPLETED")) {
                System.out.println("Finished exporting endpoints.");
            } else {
                System.err.println("Failed to export endpoints.");
                System.exit(1);
            }
        } catch (PinpointException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Download files from an Amazon S3 bucket and write them to the path location.
    public static void downloadFromS3(S3Client s3Client, String path, String
s3BucketName, List<String> objectKeys) {

        String newPath;
        try {
            for (String key : objectKeys) {
                GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .bucket(s3BucketName)
                    .key(key)
                    .build();

                ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
                byte[] data = objectBytes.asByteArray();

                // Write the data to a local file.
                String fileSuffix = new
SimpleDateFormat("yyyyMMddHHmmss").format(new Date());
                newPath = path + fileSuffix+".gz";
                File myFile = new File(newPath);
                OutputStream os = new FileOutputStream(myFile);
                os.write(data);
            }
            System.out.println("Download finished.");
        }
    }
}

```

```
        } catch (S3Exception | NullPointerException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see [CreateExportJob](#) in *AWS SDK for Java 2.x API Reference*.

## Display information about an existing Amazon Pinpoint endpoint

The following code examples show how to get endpoints.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void lookupPinpointEndpoint(PinpointClient pinpoint, String
appId, String endpoint ) {

    try {
        GetEndpointRequest appRequest = GetEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpoint)
            .build();

        GetEndpointResponse result = pinpoint.getEndpoint(appRequest);
        EndpointResponse endResponse = result.endpointResponse();

        // Uses the Google Gson library to pretty print the endpoint JSON.
        Gson gson = new GsonBuilder()
            .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
            .setPrettyPrinting()
            .create();

        String endpointJson = gson.toJson(endResponse);
        System.out.println(endpointJson);

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

- For API details, see [GetEndpoint](#) in *AWS SDK for Java 2.x API Reference*.

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun lookupPinpointEndpoint(appId: String?, endpoint: String?) {  
  
    PinpointClient { region = "us-west-2" }.use { pinpoint ->  
        val result = pinpoint.getEndpoint(  
            GetEndpointRequest {  
                applicationId = appId  
                endpointId = endpoint  
            }  
        )  
        val endResponse = result.endpointResponse  
  
        // Uses the Google Gson library to pretty print the endpoint JSON.  
        val gson: com.google.gson.Gson = GsonBuilder()  
            .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)  
            .setPrettyPrinting()  
            .create()  
  
        val endpointJson: String = gson.toJson(endResponse)  
        println(endpointJson)  
    }  
}
```

- For API details, see [GetEndpoint](#) in *AWS SDK for Kotlin API reference*.

## Import an Amazon Pinpoint segment

The following code example shows how to import a segment.

Java

### SDK for Java 2.x

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Import a segment.

```
public static ImportJobResponse createImportSegment(PinpointClient client,  
                                                    String appId,  
                                                    String bucket,  
                                                    String key,  
                                                    String roleArn) {  
  
    try {  
        ImportJobRequest importRequest = ImportJobRequest.builder()  
            .defineSegment(true)  
            .registerEndpoints(true)  
            .roleArn(roleArn)  
            .format(Format.JSON)  
            .s3Url("s3://" + bucket + "/" + key)  
            .build();  
  
        CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()  
            .importJobRequest(importRequest)
```

```
        .applicationId(appId)
        .build();

        CreateImportJobResponse jobResponse =
client.createImportJob(jobRequest);
        return jobResponse.importJobResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}
```

- For API details, see [CreateImportJob](#) in *AWS SDK for Java 2.x API Reference*.

## List Amazon Pinpoint endpoints that are associated with a specific user ID

The following code example shows how to list endpoints.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllEndpoints(PinpointClient pinpoint,
                                    String applicationId,
                                    String userId) {

    try {
        GetUserEndpointsRequest endpointsRequest =
GetUserEndpointsRequest.builder()
        .userId(userId)
        .applicationId(applicationId)
        .build();

        GetUserEndpointsResponse response =
pinpoint.getUserEndpoints(endpointsRequest);
        List<EndpointResponse> endpoints = response.endpointsResponse().item();

        // Display the results.
        for (EndpointResponse endpoint: endpoints) {
            System.out.println("The channel type is: "+endpoint.channelType());
            System.out.println("The address is "+endpoint.address());
        }
    } catch ( PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see  [GetUserEndpoints](#) in *AWS SDK for Java 2.x API Reference*.

## List Amazon Pinpoint segments in an application

The following code examples show how to list segments.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List segments.

```
public static void listSegs( PinpointClient pinpoint, String appId) {

    try {
        GetSegmentsRequest request = GetSegmentsRequest.builder()
            .applicationId(appId)
            .build();

        GetSegmentsResponse response = pinpoint.getSegments(request);
        List<SegmentResponse> segments = response.segmentsResponse().item();
        for(SegmentResponse segment: segments) {
            System.out.println("Segement " + segment.id() + " " +
segment.name() + " " + segment.lastModifiedDate());
        }
    } catch ( PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [GetSegments](#) in *AWS SDK for Java 2.x API Reference*.

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listSegs(appId: String?) {

    PinpointClient { region = "us-west-2" }.use { pinpoint ->

        val response = pinpoint.getSegments(
            GetSegmentsRequest {
                applicationId = appId
            }
        )
        response.segmentsResponse?.item?.forEach { segment ->
            println("Segement id is ${segment.id}")
        }
    }
}
```

```
        }  
    }  
}
```

- For API details, see [GetSegments](#) in *AWS SDK for Kotlin API reference*.

## Send email and text messages with Amazon Pinpoint using an AWS SDK

The following code examples show how to send email and text messages with Amazon Pinpoint.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Send an email message.

```
public static void sendEmail(PinpointClient pinpoint,  
                             String subject,  
                             String appId,  
                             String senderAddress,  
                             String toAddress) {  
  
    try {  
        Map<String,AddressConfiguration> addressMap = new HashMap<>();  
        AddressConfiguration configuration = AddressConfiguration.builder()  
            .channelType(ChannelType.EMAIL)  
            .build();  
  
        addressMap.put(toAddress, configuration);  
        SimpleEmailPart emailPart = SimpleEmailPart.builder()  
            .data(htmlBody)  
            .charset(charset)  
            .build() ;  
  
        SimpleEmailPart subjectPart = SimpleEmailPart.builder()  
            .data(subject)  
            .charset(charset)  
            .build() ;  
  
        SimpleEmail simpleEmail = SimpleEmail.builder()  
            .htmlPart(emailPart)  
            .subject(subjectPart)  
            .build();  
  
        EmailMessage emailMessage = EmailMessage.builder()  
            .body(htmlBody)  
            .fromAddress(senderAddress)  
            .simpleEmail(simpleEmail)  
            .build();  
  
        DirectMessageConfiguration directMessageConfiguration =  
        DirectMessageConfiguration.builder()  
            .emailMessage(emailMessage)  
            .build();
```

```
MessageRequest messageRequest = MessageRequest.builder()
    .addresses(addressMap)
    .messageConfiguration(directMessageConfiguration)
    .build();

SendMessagesRequest messagesRequest = SendMessagesRequest.builder()
    .applicationId(appId)
    .messageRequest(messageRequest)
    .build();

pinpoint.sendMessages(messagesRequest);

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Send an SMS message.

```
public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId, String originationNumber, String destinationNumber) {

    try {
        Map<String, AddressConfiguration> addressMap = new HashMap<String,
AddressConfiguration>();
        AddressConfiguration addConfig = AddressConfiguration.builder()
            .channelType(ChannelType.SMS)
            .build();

        addressMap.put(destinationNumber, addConfig);
        SMSMessage smsMessage = SMSMessage.builder()
            .body(message)
            .messageType(messageType)
            .originationNumber(originationNumber)
            .senderId(senderId)
            .keyword(registeredKeyword)
            .build();

        // Create a DirectMessageConfiguration object.
        DirectMessageConfiguration direct =
        DirectMessageConfiguration.builder()
            .smsMessage(smsMessage)
            .build();

        MessageRequest msgReq = MessageRequest.builder()
            .addresses(addressMap)
            .messageConfiguration(direct)
            .build();

        // create a SendMessagesRequest object
        SendMessagesRequest request = SendMessagesRequest.builder()
            .applicationId(appId)
            .messageRequest(msgReq)
            .build();

        SendMessagesResponse response= pinpoint.sendMessages(request);
        MessageResponse msg1 = response.messageResponse();
        Map map1 = msg1.result();

        //Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println(k + ":" + v));

    } catch (PinpointException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Send batch SMS messages.

```
public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId, String originationNumber, String destinationNumber, String
destinationNumber1) {
    try {
        Map<String, AddressConfiguration> addressMap = new HashMap<String,
AddressConfiguration>();
        AddressConfiguration addConfig = AddressConfiguration.builder()
            .channelType(ChannelType.SMS)
            .build();

        // Add an entry to the Map object for each number to whom you want to
send a message.
        addressMap.put(destinationNumber, addConfig);
        addressMap.put(destinationNumber1, addConfig);
        SMSMessage smsMessage = SMSMessage.builder()
            .body(message)
            .messageType(messageType)
            .originationNumber(originationNumber)
            .senderId(senderId)
            .keyword(registeredKeyword)
            .build();

        // Create a DirectMessageConfiguration object.
        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
            .smsMessage(smsMessage)
            .build();

        MessageRequest msgReq = MessageRequest.builder()
            .addresses(addressMap)
            .messageConfiguration(direct)
            .build();

        // Create a SendMessagesRequest object.
        SendMessagesRequest request = SendMessagesRequest.builder()
            .applicationId(appId)
            .messageRequest(msgReq)
            .build();

        SendMessagesResponse response= pinpoint.sendMessages(request);
        MessageResponse msg1 = response.messageResponse();
        Map map1 = msg1.result();

        // Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SendMessages](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { PinpointClient } from "@aws-sdk/client-pinpoint";
// Set the AWS Region.
const REGION = "us-east-1";
//Set the MediaConvert Service Object
const pinClient = new PinpointClient({region: REGION});
export { pinClient };
```

Send an email message.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessagesCommand } from "@aws-sdk/client-pinpoint";
import { pinClient } from "./libs/pinClient.js";

// The FromAddress must be verified in SES.
const fromAddress = "FROM_ADDRESS";
const toAddress = "TO_ADDRESS";
const projectId = "PINPOINT_PROJECT_ID";

// The subject line of the email.
var subject = "Amazon Pinpoint Test (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
var body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)
-----
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in
Node.js.
For more information, see https://aws.amazon.com/sdk-for-node-js/`;

// The body of the email for recipients whose email clients support HTML content.
var body_html = `<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint Email API</a>
  using the
    <a href='https://aws.amazon.com/sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js</a>.</p>
</body>
</html>`;

// The character encoding for the subject line and message body of the email.
var charset = "UTF-8";

const params = {
  ApplicationId: projectId,
  MessageRequest: {
    Addresses: {
      [toAddress]: {
        ChannelType: "EMAIL",
      },
    },
    MessageConfiguration: {
```

```

EmailMessage: {
  FromAddress: fromAddress,
  SimpleEmail: {
    Subject: {
      Charset: charset,
      Data: subject,
    },
    HtmlPart: {
      Charset: charset,
      Data: body_html,
    },
    TextPart: {
      Charset: charset,
      Data: body_text,
    },
  },
},
};

const run = async () => {
  try {
    const data = await pinClient.send(new SendMessagesCommand(params));

    const {
      MessageResponse: { Result },
    } = data;

    const recipientResult = Result[toAddress];

    if (recipientResult.StatusCode !== 200) {
      throw new Error(recipientResult.StatusMessage);
    } else {
      console.log(recipientResult.MessageId);
    }
  } catch (err) {
    console.log(err.message);
  }
};

run();

```

Send an SMS message.

```

// Import required AWS SDK clients and commands for Node.js
import { SendMessagesCommand } from "@aws-sdk/client-pinpoint";
import { pinClient } from "./libs/pinClient.js";

("use strict");

/* The phone number or short code to send the message from. The phone number
or short code that you specify has to be associated with your Amazon Pinpoint
account. For best results, specify long codes in E.164 format. */
const originationNumber = "SENDER_NUMBER"; //e.g., +1XXXXXXXXXX

// The recipient's phone number. For best results, you should specify the phone
// number in E.164 format.
const destinationNumber = "RECEIVER_NUMBER"; //e.g., +1XXXXXXXXXX

// The content of the SMS message.
const message =
  "This message was sent through Amazon Pinpoint " +

```

```
"using the AWS SDK for JavaScript in Node.js. Reply STOP to " +  
"opt out.";  
  
/*The Amazon Pinpoint project/application ID to use when you send this message.  
Make sure that the SMS channel is enabled for the project or application  
that you choose.*/  
const projectId = "PINPOINT_PROJECT_ID"; //e.g., XXXXXXXX66e4e9986478cXXXXXXXXX  
  
/* The type of SMS message that you want to send. If you plan to send  
time-sensitive content, specify TRANSACTIONAL. If you plan to send  
marketing-related content, specify PROMOTIONAL.*/  
var messageType = "TRANSACTIONAL";  
  
// The registered keyword associated with the originating short code.  
var registeredKeyword = "myKeyword";  
  
/* The sender ID to use when sending the message. Support for sender ID  
// varies by country or region. For more information, see  
https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-  
countries.html.*/  
  
var senderId = "MySenderId";  
  
// Specify the parameters to pass to the API.  
var params = {  
    ApplicationId: projectId,  
    MessageRequest: {  
        Addresses: {  
            [destinationNumber]: {  
                ChannelType: "SMS",  
            },  
        },  
        MessageConfiguration: {  
            SMSMessage: {  
                Body: message,  
                Keyword: registeredKeyword,  
                MessageType: messageType,  
                OriginationNumber: originationNumber,  
                SenderId: senderId,  
            },  
        },  
    },  
};  
  
const run = async () => {  
    try {  
        const data = await pinClient.send(new SendMessagesCommand(params));  
        return data; // For unit tests.  
        console.log(  
            "Message sent! " +  
            data["MessageResponse"]["Result"][destinationNumber]["StatusMessage"]  
        );  
    } catch (err) {  
        console.log(err);  
    }  
};  
run();
```

- For API details, see [SendMessages](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Send an email message.

```
'use strict';

const AWS = require('aws-sdk');

// The AWS Region that you want to use to send the email. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/
const aws_region = "us-west-2"

// The "From" address. This address has to be verified in Amazon Pinpoint
// in the region that you use to send email.
const senderAddress = "sender@example.com";

// The address on the "To" line. If your Amazon Pinpoint account is in
// the sandbox, this address also has to be verified.
var toAddress = "recipient@example.com";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
const appId = "ce796be37f32f178af652b26eexample";

// The subject line of the email.
var subject = "Amazon Pinpoint (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
var body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)
-----
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in
Node.js.
For more information, see https:\`/aws.amazon.com/sdk-for-node-js\`;

// The body of the email for recipients whose email clients support HTML content.
var body_html = `<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint API</a> using
    the
    <a href='https://aws.amazon.com/sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js</a>.</p>
</body>
</html>`;

// The character encoding the you want to use for the subject line and
// message body of the email.
var charset = "UTF-8";

// Specify that you're using a shared credentials file.
var credentials = new AWS.SharedIniFileCredentials({profile: 'default'});
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({region:aws_region});

//Create a new Pinpoint object.
var pinpoint = new AWS.Pinpoint();

// Specify the parameters to pass to the API.
var params = {
  ApplicationId: appId,
```

```

MessageRequest: {
    Addresses: {
        [toAddress]:{
            ChannelType: 'EMAIL'
        }
    },
    MessageConfiguration: {
        EmailMessage: {
            FromAddress: senderAddress,
            SimpleEmail: {
                Subject: {
                    Charset: charset,
                    Data: subject
                },
                HtmlPart: {
                    Charset: charset,
                    Data: body_html
                },
                TextPart: {
                    Charset: charset,
                    Data: body_text
                }
            }
        }
    }
};

//Try to send the email.
pinpoint.sendMessages(params, function(err, data) {
    // If something goes wrong, print an error message.
    if(err) {
        console.log(err.message);
    } else {
        console.log("Email sent! Message ID: ", data['MessageResponse']['Result'][toAddress]['MessageId']);
    }
});

```

Send an SMS message.

```

'use strict';

var AWS = require('aws-sdk');

// The AWS Region that you want to use to send the message. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/.
var aws_region = "us-east-1";

// The phone number or short code to send the message from. The phone number
// or short code that you specify has to be associated with your Amazon Pinpoint
// account. For best results, specify long codes in E.164 format.
var originationNumber = "+12065550199";

// The recipient's phone number. For best results, you should specify the
// phone number in E.164 format.
var destinationNumber = "+14255550142";

// The content of the SMS message.
var message = "This message was sent through Amazon Pinpoint "
    + "using the AWS SDK for JavaScript in Node.js. Reply STOP to "

```

```
+ "opt out.";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
var applicationId = "ce796be37f32f178af652b26eexample";

// The type of SMS message that you want to send. If you plan to send
// time-sensitive content, specify TRANSACTIONAL. If you plan to send
// marketing-related content, specify PROMOTIONAL.
var messageType = "TRANSACTIONAL";

// The registered keyword associated with the originating short code.
var registeredKeyword = "myKeyword";

// The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
// https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
// countries.html
var senderId = "MySenderId";

// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({profile: 'default'});
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({region:aws_region});

//Create a new Pinpoint object.
var pinpoint = new AWS.Pinpoint();

// Specify the parameters to pass to the API.
var params = {
    ApplicationId: applicationId,
    MessageRequest: {
        Addresses: {
            [destinationNumber]: {
                ChannelType: 'SMS'
            }
        },
        MessageConfiguration: {
            SMSMessage: {
                Body: message,
                Keyword: registeredKeyword,
                MessageType: messageType,
                OriginationNumber: originationNumber,
                SenderId: senderId,
            }
        }
    }
};

//Try to send the message.
pinpoint.sendMessages(params, function(err, data) {
    // If something goes wrong, print an error message.
    if(err) {
        console.log(err.message);
    // Otherwise, show the unique ID for the message.
    } else {
        console.log("Message sent! "
            + data['MessageResponse']['Result'][destinationNumber]['StatusMessage']);
    }
});
```

- For API details, see [SendMessages](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun sendEmail(  
    msgSubject: String?,  
    appId: String?,  
    senderAddress: String?,  
    toAddress: String  
) {  
  
    // The body of the email for recipients whose email clients support HTML  
    // content.  
    val htmlBody = (  
        "<h1>Amazon Pinpoint test (AWS SDK for Kotlin)</h1>" +  
        "<p>This email was sent through the <a href='https://aws.amazon.com/  
pinpoint/'>" +  
        "Amazon Pinpoint</a> Email API"  
    )  
  
    // The character encoding to use for the subject line and the message body.  
    val charsetVal = "UTF-8"  
  
    val addressMap = mutableMapOf<String, AddressConfiguration>()  
    val configuration = AddressConfiguration {  
        channelType = ChannelType.Email  
    }  
  
    addressMap[toAddress] = configuration  
    val emailPart = SimpleEmailPart {  
        data = htmlBody  
        charset = charsetVal  
    }  
  
    val subjectPart0b = SimpleEmailPart {  
        data = msgSubject  
        charset = charsetVal  
    }  
  
    val simpleEmail0b = SimpleEmail {  
        htmlPart = emailPart  
        subject = subjectPart0b  
    }  
  
    val emailMessage0b = EmailMessage {  
        body = htmlBody  
        fromAddress = senderAddress  
        simpleEmail = simpleEmail0b  
    }  
  
    val directMessageConfiguration0b = DirectMessageConfiguration {
```

```
        emailMessage = emailMessage0b
    }

    val messageRequest0b = MessageRequest {
        addresses = addressMap
        messageConfiguration = directMessageConfiguration0b
    }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        pinpoint.sendMessages(
            SendMessagesRequest {
                applicationId = appId
                messageRequest = messageRequest0b
            }
        )
        println("The email message was successfully sent")
    }
}
```

- For API details, see [SendMessages](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Send an email message.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def send_email_message(
    pinpoint_client, app_id, sender, to_addresses, char_set, subject,
    html_message, text_message):
    """
    Sends an email message with HTML and plain text versions.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param app_id: The Amazon Pinpoint project ID to use when you send this
    message.
    :param sender: The "From" address. This address must be verified in
        Amazon Pinpoint in the AWS Region you're using to send email.
    :param to_addresses: The addresses on the "To" line. If your Amazon Pinpoint
    account
        is in the sandbox, these addresses must be verified.
    :param char_set: The character encoding to use for the subject line and message
    body of the email.
    :param subject: The subject line of the email.
    :param html_message: The body of the email for recipients whose email clients
    can
        display HTML content.
    :param text_message: The body of the email for recipients whose email clients
    don't support HTML content.
    :return: A dict of to_addresses and their message IDs.
    """

```

```

try:
    response = pinpoint_client.send_messages(
        ApplicationId=app_id,
        MessageRequest={
            'Addresses': [
                to_address: {'ChannelType': 'EMAIL'} for to_address in
to_addresses
            ],
            'MessageConfiguration': {
                'EmailMessage': {
                    'FromAddress': sender,
                    'SimpleEmail': {
                        'Subject': {'Charset': char_set, 'Data': subject},
                        'HtmlPart': {'Charset': char_set, 'Data':
html_message},
                        'TextPart': {'Charset': char_set, 'Data':
text_message}}}}})
    except ClientError:
        logger.exception("Couldn't send email.")
        raise
else:
    return {
        to_address: message['MessageId'] for
        to_address, message in response['MessageResponse']['Result'].items()
    }

def main():
    app_id = "ce796be37f32f178af652b26eexample"
    sender = "sender@example.com"
    to_address = "recipient@example.com"
    char_set = "UTF-8"
    subject = "Amazon Pinpoint Test (SDK for Python (Boto3))"
    text_message = """Amazon Pinpoint Test (SDK for Python)
-----
This email was sent with Amazon Pinpoint using the AWS SDK for Python (Boto3).
For more information, see https://aws.amazon.com/sdk-for-python/
"""
    html_message = """<html>
<head></head>
<body>
    <h1>Amazon Pinpoint Test (SDK for Python (Boto3))</h1>
    <p>This email was sent with
        <a href='https://aws.amazon.com/pinpoint/'>Amazon Pinpoint</a> using the
        <a href='https://aws.amazon.com/sdk-for-python/'>
            AWS SDK for Python (Boto3)</a>.</p>
</body>
</html>
"""

    print("Sending email.")
    message_ids = send_email_message(
        boto3.client('pinpoint'), app_id, sender, [to_address], char_set, subject,
        html_message, text_message)
    print(f"Message sent! Message IDs: {message_ids}")

if __name__ == '__main__':
    main()

```

Send an SMS message.

```
import logging
```

```

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def send_sms_message(
    pinpoint_client, app_id, origination_number, destination_number, message,
    message_type):
    """
    Sends an SMS message with Amazon Pinpoint.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param app_id: The Amazon Pinpoint project/application ID to use when you send
        this message. The SMS channel must be enabled for the project or
        application.
    :param destination_number: The recipient's phone number in E.164 format.
    :param origination_number: The phone number to send the message from. This
        phone
            number must be associated with your Amazon Pinpoint
        account and be in E.164 format.
    :param message: The content of the SMS message.
    :param message_type: The type of SMS message that you want to send. If you send
        time-sensitive content, specify TRANSACTIONAL. If you send
        marketing-related content, specify PROMOTIONAL.
    :return: The ID of the message.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=app_id,
            MessageRequest={
                'Addresses': {destination_number: {'ChannelType': 'SMS'}},
                'MessageConfiguration': {
                    'SMSMessage': {
                        'Body': message,
                        'MessageType': message_type,
                        'OriginationNumber': origination_number}}})
    except ClientError:
        logger.exception("Couldn't send message.")
        raise
    else:
        return response['MessageResponse']['Result'][destination_number]
        ['MessageId']

def main():
    app_id = "ce796be37f32f178af652b26eexample"
    origination_number = "+12065550199"
    destination_number = "+14255550142"
    message = (
        "This is a sample message sent from Amazon Pinpoint by using the AWS SDK"
    for "
        "Python (Boto 3).")
    message_type = "TRANSACTIONAL"

    print("Sending SMS message.")
    message_id = send_sms_message(
        boto3.client('pinpoint'), app_id, origination_number, destination_number,
        message, message_type)
    print(f"Message sent! Message ID: {message_id}.")

if __name__ == '__main__':
    main()

```

- For API details, see [SendMessages](#) in *AWS SDK for Python (Boto3) API Reference*.

## Send templated email and text messages with Amazon Pinpoint using an AWS SDK

The following code example shows how to send templated email and text messages with Amazon Pinpoint.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Send an email message with an existing email template.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def send_templated_email_message(
    pinpoint_client, project_id, sender, to_addresses, template_name,
    template_version):
    """
    Sends an email message with HTML and plain text versions.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param project_id: The Amazon Pinpoint project ID to use when you send this
        message.
    :param sender: The "From" address. This address must be verified in
        Amazon Pinpoint in the AWS Region you're using to send email.
    :param to_addresses: The addresses on the "To" line. If your Amazon Pinpoint
        account is in the sandbox, these addresses must be
        verified.
    :param template_name: The name of the email template to use when sending the
        message.
    :param template_version: The version number of the message template.

    :return: A dict of to_addresses and their message IDs.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=project_id,
            MessageRequest={
                'Addresses': {
                    'to_address': [
                        {'ChannelType': 'EMAIL'}
                    ] for to_address in to_addresses
                },
                'MessageConfiguration': {
                    'EmailMessage': {'FromAddress': sender}
                },
                'TemplateConfiguration': {
                    'EmailTemplate': {
                        'Name': template_name,
                        'Version': template_version
                    }
                }
            }
        )
        return response['Messages']
    except ClientError as e:
        logger.error(e)
        raise e
```

```

        }
    )
except ClientError:
    logger.exception("Couldn't send email.")
    raise
else:
    return {
        to_address: message['MessageId'] for
        to_address, message in response['MessageResponse']['Result'].items()
    }

def main():
    project_id = "296b04b342374fceb661bf494example"
    sender = "sender@example.com"
    to_addresses = ["recipient@example.com"]
    template_name = "My_Email_Template"
    template_version = "1"

    print("Sending email.")
    message_ids = send templated_email_message(
        boto3.client('pinpoint'), project_id, sender, to_addresses, template_name,
        template_version)
    print(f"Message sent! Message IDs: {message_ids}")

if __name__ == '__main__':
    main()

```

Send a text message with an existing SMS template.

```

import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def send templated_sms_message(
    pinpoint_client,
    project_id,
    destination_number,
    message_type,
    origination_number,
    template_name,
    template_version):
    """
    Sends an SMS message to a specific phone number using a pre-defined template.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param project_id: An Amazon Pinpoint project (application) ID.
    :param destination_number: The phone number to send the message to.
    :param message_type: The type of SMS message (promotional or transactional).
    :param origination_number: The phone number that the message is sent from.
    :param template_name: The name of the SMS template to use when sending the
    message.
    :param template_version: The version number of the message template.

    :return The ID of the message.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=project_id,

```

```
MessageRequest={  
    'Addresses': {  
        'destination_number': {  
            'ChannelType': 'SMS'  
        }  
    },  
    'MessageConfiguration': {  
        'SMSMessage': {  
            'MessageType': message_type,  
            'OriginationNumber': origination_number  
        }  
    },  
    'TemplateConfiguration': {  
        'SMSTemplate': {  
            'Name': template_name,  
            'Version': template_version  
        }  
    }  
}  
)  
  
except ClientError:  
    logger.exception("Couldn't send message.")  
    raise  
else:  
    return response['MessageResponse']['Result'][destination_number]  
['MessageId']  
  
def main():  
    region = "us-east-1"  
    origination_number = "+18555550001"  
    destination_number = "+14255550142"  
    project_id = "7353f53e6885409fa32d07cedexample"  
    message_type = "TRANSACTIONAL"  
    template_name = "My_SMS_Template"  
    template_version = "1"  
    message_id = send_templated_sms_message(  
        boto3.client('pinpoint', region_name=region), project_id,  
        destination_number, message_type, origination_number, template_name,  
        template_version)  
    print(f"Message sent! Message ID: {message_id}.")  
  
if __name__ == '__main__':  
    main()
```

- For API details, see [SendMessages](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update an Amazon Pinpoint endpoint

The following code example shows how to update an endpoint.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static EndpointResponse createEndpoint(PinpointClient client, String
appId) {
    String endpointId = UUID.randomUUID().toString();
    System.out.println("Endpoint ID: " + endpointId);

    try {
        EndpointRequest endpointRequest = createEndpointRequestData();
        UpdateEndpointRequest updateEndpointRequest =
UpdateEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpointId)
            .endpointRequest(endpointRequest)
            .build();

        UpdateEndpointResponse updateEndpointResponse =
client.updateEndpoint(updateEndpointRequest);
        System.out.println("Update Endpoint Response: " +
updateEndpointResponse.messageBody());

        GetEndpointRequest getEndpointRequest = GetEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpointId)
            .build();

        GetEndpointResponse getEndpointResponse =
client.getEndpoint(getEndpointRequest);
        System.out.println(getEndpointResponse.endpointResponse().address());

        System.out.println(getEndpointResponse.endpointResponse().channelType());
        System.out.println(getEndpointResponse.endpointResponse().applicationId());
        System.out.println(getEndpointResponse.endpointResponse().endpointStatus());
        System.out.println(getEndpointResponse.endpointResponse().requestId());
        System.out.println(getEndpointResponse.endpointResponse().user());

        return getEndpointResponse.endpointResponse();
    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

private static EndpointRequest createEndpointRequestData() {

    try {
        List<String> favoriteTeams = new ArrayList<>();
        favoriteTeams.add("Lakers");
        favoriteTeams.add("Warriors");
        HashMap<String, List<String>> customAttributes = new HashMap<>();
        customAttributes.put("team", favoriteTeams);

        EndpointDemographic demographic = EndpointDemographic.builder()
            .appVersion("1.0")
            .make("apple")
            .model("iPhone")
            .modelVersion("7")
            .platform("ios")
            .platformVersion("10.1.1")
            .timezone("America/Los_Angeles")
            .build();

        EndpointLocation location = EndpointLocation.builder()
```

```
.city("Los Angeles")
.country("US")
.latitude(34.0)
.longitude(-118.2)
.postalCode("90068")
.region("CA")
.build();

Map<String,Double> metrics = new HashMap<>();
metrics.put("health", 100.00);
metrics.put("luck", 75.00);

EndpointUser user = EndpointUser.builder()
    .userId(UUID.randomUUID().toString())
    .build();

DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm'Z'"); // Quoted "Z" to indicate UTC, no timezone offset
String nowAsISO = df.format(new Date());

return EndpointRequest.builder()
    .address(UUID.randomUUID().toString())
    .attributes(customAttributes)
    .channelType("APNS")
    .demographic(demographic)
    .effectiveDate(nowAsISO)
    .location(location)
    .metrics(metrics)
    .optOut("NONE")
    .requestId(UUID.randomUUID().toString())
    .user(user)
    .build();

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}
```

- For API details, see [UpdateEndpoint](#) in *AWS SDK for Java 2.x API Reference*.

## Update an Amazon Pinpoint channel

The following code example shows how to update channels.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
private static SMSChannelResponse getSMSChannel(PinpointClient client, String appId) {

    try {
        GetSmsChannelRequest request = GetSmsChannelRequest.builder()
            .applicationId(appId)
```

```
.build();

SMSChannelResponse response =
client.getSmsChannel(request).smsChannelResponse();
System.out.println("Channel state is " + response.enabled());
return response;

} catch ( PinpointException e ) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}

private static void toggleSmsChannel(PinpointClient client, String appId,
SMSChannelResponse getResponse) {
    boolean enabled = !getResponse.enabled();

    try {
        SMSChannelRequest request = SMSChannelRequest.builder()
            .enabled(enabled)
            .build();

        UpdateSmsChannelRequest updateRequest =
UpdateSmsChannelRequest.builder()
    .smsChannelRequest(request)
    .applicationId(appId)
    .build();

        UpdateSmsChannelResponse result =
client.updateSmsChannel(updateRequest);
        System.out.println("Channel state: " +
result.smsChannelResponse().enabled());

    } catch ( PinpointException e ) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [GetSmsChannel](#) in *AWS SDK for Java 2.x API Reference*.

## Code examples for Amazon Pinpoint SMS and Voice API using AWS SDKs

The following code examples show how to use Amazon Pinpoint SMS and Voice API with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon Pinpoint SMS and Voice API using AWS SDKs \(p. 1407\)](#)
  - [Send a voice message with Amazon Pinpoint SMS and Voice API using an AWS SDK \(p. 1407\)](#)

# Actions for Amazon Pinpoint SMS and Voice API using AWS SDKs

The following code examples show how to use Amazon Pinpoint SMS and Voice API with AWS SDKs. Each example calls an individual service function.

## Examples

- [Send a voice message with Amazon Pinpoint SMS and Voice API using an AWS SDK \(p. 1407\)](#)

## Send a voice message with Amazon Pinpoint SMS and Voice API using an AWS SDK

The following code examples show how to send a voice message with Amazon Pinpoint SMS and Voice API.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void sendVoiceMsg(PinpointSmsVoiceClient client, String originationNumber, String destinationNumber) {

    try {
        SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
            .languageCode(languageCode)
            .text(ssmlMessage)
            .voiceId(voiceName)
            .build();

        VoiceMessageContent content = VoiceMessageContent.builder()
            .ssmlMessage(ssmlMessageType)
            .build();

        SendVoiceMessageRequest voiceMessageRequest =
        SendVoiceMessageRequest.builder()
            .destinationPhoneNumber(destinationNumber)
            .originationPhoneNumber(originationNumber)
            .content(content)
            .build();

        client.sendVoiceMessage(voiceMessageRequest);
        System.out.println("The message was sent successfully.");

    } catch (PinpointSmsVoiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SendVoiceMessage](#) in [AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
'use strict'

var AWS = require('aws-sdk');

// The AWS Region that you want to use to send the voice message. For a list of
// AWS Regions where the Amazon Pinpoint SMS and Voice API is available, see
// https://docs.aws.amazon.com/pinpoint-sms-voice/latest/APIReference/
var aws_region = "us-east-1";

// The phone number that the message is sent from. The phone number that you
// specify has to be associated with your Amazon Pinpoint account. For best
// results, you
// should specify the phone number in E.164 format.
var originationNumber = "+12065550110";

// The recipient's phone number. For best results, you should specify the phone
// number in E.164 format.
var destinationNumber = "+12065550142";

// The language to use when sending the message. For a list of supported
// languages, see https://docs.aws.amazon.com/polly/latest/dg/
SupportedLanguage.html
var languageCode = "en-US";

// The Amazon Polly voice that you want to use to send the message. For a list
// of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
var voiceId = "Matthew";

// The content of the message. This example uses SSML to customize and control
// certain aspects of the message, such as the volume or the speech rate.
// The message can't contain any line breaks.
var ssmlMessage = "<speak>
  + "This is a test message sent from <emphasis>Amazon Pinpoint</emphasis> "
  + "using the <break strength='weak' />AWS SDK for JavaScript in Node.js. "
  + "<amazon:effect phonation='soft'>Thank you for listening."
  + "</amazon:effect>"
  + "</speak>";

// The phone number that you want to appear on the recipient's device. The phone
// number that you specify has to be associated with your Amazon Pinpoint account.
var callerId = "+12065550199";

// The configuration set that you want to use to send the message.
var configurationSet = "ConfigSet";

// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({profile: 'default'});
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({region:aws_region});

//Create a new Pinpoint object.
var pinpointSMSvoice = new AWS.PinpointSMSVoice();
```

```
var params = {
  CallerId: callerId,
  ConfigurationSetName: configurationSet,
  Content: {
    SSMLMessage: {
      LanguageCode: languageCode,
      Text: ssmlMessage,
      VoiceId: voiceId
    }
  },
  DestinationPhoneNumber: destinationNumber,
  OriginationPhoneNumber: originationNumber
};

//Try to send the message.
pinpointsmsvoice.sendVoiceMessage(params, function(err, data) {
  // If something goes wrong, print an error message.
  if(err) {
    console.log(err.message);
  // Otherwise, show the unique ID for the message.
  } else {
    console.log("Message sent! Message ID: " + data['MessageId']);
  }
});
```

- For API details, see [SendVoiceMessage](#) in *AWS SDK for JavaScript API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def send_voice_message(
    sms_voice_client, origination_number, caller_id, destination_number,
    language_code, voice_id, ssml_message):
    """
    Sends a voice message using speech synthesis provided by Amazon Polly.

    :param sms_voice_client: A Boto3 PinpointSMSVoice client.
    :param origination_number: The phone number that the message is sent from.
        The phone number must be associated with your Amazon Pinpoint account and be in E.164 format.
    :param caller_id: The phone number that you want to appear on the recipient's device. The phone number must be associated with your Amazon Pinpoint account and be in E.164 format.
    :param destination_number: The recipient's phone number. Specify the phone number in E.164 format.
    :param language_code: The language to use when sending the message.
    """
```

```
:param voice_id: The Amazon Polly voice that you want to use to send the
message.
:param ssml_message: The content of the message. This example uses SSML to
control
    certain aspects of the message, such as the volume and the
    speech rate. The message must not contain line breaks.
:return: The ID of the message.
"""
try:
    response = sms_voice_client.send_voice_message(
        DestinationPhoneNumber=destination_number,
        OriginationPhoneNumber=origination_number,
        CallerId=caller_id,
        Content={
            'SSMLMessage': {
                'LanguageCode': language_code,
                'VoiceId': voice_id,
                'Text': ssml_message}})
except ClientError:
    logger.exception(
        "Couldn't send message from %s to %s.", origination_number,
        destination_number)
    raise
else:
    return response['MessageId']

def main():
    origination_number = "+12065550110"
    caller_id = "+12065550199"
    destination_number = "+12065550142"
    language_code = "en-US"
    voice_id = "Matthew"
    ssml_message = (
        "<speak>"
        "This is a test message sent from <emphasis>Amazon Pinpoint</emphasis> "
        "using the <break strength='weak'>/>AWS SDK for Python (Boto3). "
        "<amazon:effect phonation='soft'>Thank you for listening."
        "</amazon:effect>"
        "</speak>")
    print(f"Sending voice message from {origination_number} to
{destination_number}.")
    message_id = send_voice_message(
        boto3.client('pinpoint-sms-voice'), origination_number, caller_id,
        destination_number, language_code, voice_id, ssml_message)
    print(f"Message sent!\nMessage ID: {message_id}")

if __name__ == '__main__':
    main()
```

- For API details, see [SendVoiceMessage](#) in *AWS SDK for Python (Boto3) API Reference*.

## Code examples for Amazon Polly using AWS SDKs

The following code examples show how to use Amazon Polly with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon Polly using AWS SDKs \(p. 1411\)](#)
  - [Delete an Amazon Polly lexicon using an AWS SDK \(p. 1411\)](#)

- Get an Amazon Polly lexicon using an AWS SDK (p. 1412)
- Get data about an Amazon Polly speech synthesis task using an AWS SDK (p. 1414)
- Get Amazon Polly voices available for synthesis using an AWS SDK (p. 1414)
- List Amazon Polly pronunciation lexicons using an AWS SDK (p. 1417)
- Start an Amazon Polly speech synthesis task using an AWS SDK (p. 1419)
- Store an Amazon Polly pronunciation lexicon using an AWS SDK (p. 1421)
- Synthesize speech from text with Amazon Polly using an AWS SDK (p. 1423)
- Scenarios for Amazon Polly using AWS SDKs (p. 1427)
  - Create a lip-sync application with Amazon Polly using an AWS SDK (p. 1428)
- Cross-service examples for Amazon Polly using AWS SDKs (p. 1428)
  - Convert text to speech and back to text using an AWS SDK (p. 1428)

## Actions for Amazon Polly using AWS SDKs

The following code examples show how to use Amazon Polly with AWS SDKs. Each example calls an individual service function.

### Examples

- Delete an Amazon Polly lexicon using an AWS SDK (p. 1411)
- Get an Amazon Polly lexicon using an AWS SDK (p. 1412)
- Get data about an Amazon Polly speech synthesis task using an AWS SDK (p. 1414)
- Get Amazon Polly voices available for synthesis using an AWS SDK (p. 1414)
- List Amazon Polly pronunciation lexicons using an AWS SDK (p. 1417)
- Start an Amazon Polly speech synthesis task using an AWS SDK (p. 1419)
- Store an Amazon Polly pronunciation lexicon using an AWS SDK (p. 1421)
- Synthesize speech from text with Amazon Polly using an AWS SDK (p. 1423)

## Delete an Amazon Polly lexicon using an AWS SDK

The following code example shows how to delete an Amazon Polly lexicon.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class DeleteLexicon
{
    public static async Task Main()
    {
        string lexiconName = "SampleLexicon";

        var client = new AmazonPollyClient();
```

```
        var success = await DeletePollyLexiconAsync(client, lexiconName);

        if (success)
        {
            Console.WriteLine($"Successfully deleted {lexiconName}.");
        }
        else
        {
            Console.WriteLine($"Could not delete {lexiconName}.");
        }
    }

    ///<summary>
    ///<summary> Deletes the named Amazon Polly lexicon.
    ///</summary>
    ///<param name="client">The initialized Amazon Polly client object.</
param>
    ///<param name="lexiconName">The name of the Amazon Polly lexicon to
    ///<delete.</param>
    ///<returns>A Boolean value indicating the success of the operation.</
returns>
    public static async Task<bool> DeletePollyLexiconAsync(
        AmazonPollyClient client,
        string lexiconName)
    {
        var deleteLexiconRequest = new DeleteLexiconRequest()
        {
            Name = lexiconName,
        };

        var response = await client.DeleteLexiconAsync(deleteLexiconRequest);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

- For API details, see [DeleteLexicon](#) in *AWS SDK for .NET API Reference*.

## Get an Amazon Polly lexicon using an AWS SDK

The following code examples show how to get an Amazon Polly lexicon.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class GetLexicon
{
    public static async Task Main(string[] args)
    {
        string lexiconName = "SampleLexicon";
```

```
        var client = new AmazonPollyClient();

        await GetPollyLexiconAsync(client, lexiconName);
    }

    public static async Task GetPollyLexiconAsync(AmazonPollyClient client,
string lexiconName)
{
    var getLexiconRequest = new GetLexiconRequest()
    {
        Name = lexiconName,
    };

    try
    {
        var response = await client.GetLexiconAsync(getLexiconRequest);
        Console.WriteLine($"Lexicon:\n Name: {response.Lexicon.Name}");
        Console.WriteLine($"Content: {response.Lexicon.Content}");
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error: " + ex.Message);
    }
}
}
```

- For API details, see [GetLexicon in AWS SDK for .NET API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""
    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None

    def get_lexicon(self, name):
        """
        Gets metadata and contents of an existing lexicon.

        :param name: The name of the lexicon to retrieve.
        :return: The retrieved lexicon.
        """
        try:
            response = self.polly_client.get_lexicon(Name=name)
            logger.info("Got lexicon %s.", name)
        except ClientError:
            logger.exception("Couldn't get lexicon %s.", name)
```

```
        raise
    else:
        return response
```

- For API details, see [GetLexicon](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get data about an Amazon Polly speech synthesis task using an AWS SDK

The following code example shows how to get data about an existing asynchronous Amazon Polly speech synthesis task.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""
    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
        resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None

    def get_speech_synthesis_task(self, task_id):
        """
        Gets metadata about an asynchronous speech synthesis task, such as its
        status.

        :param task_id: The ID of the task to retrieve.
        :return: Metadata about the task.
        """
        try:
            response = self.polly_client.get_speech_synthesis_task(TaskId=task_id)
            task = response['SynthesisTask']
            logger.info("Got synthesis task. Status is %s.", task['TaskStatus'])
        except ClientError:
            logger.exception("Couldn't get synthesis task %s.", task_id)
            raise
        else:
            return task
```

- For API details, see [GetSpeechSynthesisTask](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get Amazon Polly voices available for synthesis using an AWS SDK

The following code examples show how to get Amazon Polly voices available for synthesis.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class DescribeVoices
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();

        var allVoicesRequest = new DescribeVoicesRequest();
        var enUsVoicesRequest = new DescribeVoicesRequest()
        {
            LanguageCode = "en-US",
        };

        try
        {
            string nextToken;
            do
            {
                var allVoicesResponse = await
client.DescribeVoicesAsync(allVoicesRequest);
                nextToken = allVoicesResponse.NextToken;
                allVoicesRequest.NextToken = nextToken;

                Console.WriteLine("\nAll voices: ");
                allVoicesResponse.Voices.ForEach(voice =>
                {
                    DisplayVoiceInfo(voice);
                });
            }
            while (nextToken is not null);

            do
            {
                var enUsVoicesResponse = await
client.DescribeVoicesAsync(enUsVoicesRequest);
                nextToken = enUsVoicesResponse.NextToken;
                enUsVoicesRequest.NextToken = nextToken;

                Console.WriteLine("\nen-US voices: ");
                enUsVoicesResponse.Voices.ForEach(voice =>
                {
                    DisplayVoiceInfo(voice);
                });
            }
            while (nextToken is not null);
        }
        catch (Exception ex)
        {
            Console.WriteLine("Exception caught: " + ex.Message);
        }
    }
}
```

```
        public static void DisplayVoiceInfo(Voice voice)
    {
        Console.WriteLine($" Name: {voice.Name}\tGender:
{voice.Gender}\tLanguageName: {voice.LanguageName}");
    }
}
```

- For API details, see [DescribeVoices](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""
    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
        resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None

    def describe.voices(self):
        """
        Gets metadata about available voices.

        :return: The list of voice metadata.
        """
        try:
            response = self.polly_client.describe.voices()
            self.voice_metadata = response['Voices']
            logger.info("Got metadata about %s voices.", len(self.voice_metadata))
        except ClientError:
            logger.exception("Couldn't get voice metadata.")
            raise
        else:
            return self.voice_metadata
```

- For API details, see [DescribeVoices](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn list_voices(client: &Client) -> Result<(), Error> {
    let resp = client.describe_VOICES().send().await?;

    println!("Voices:");

    let voices = resp.voices().unwrap_or_default();
    for voice in voices {
        println!("  Name: {}", voice.name().unwrap_or("No name!"));
        println!("  Language: {}", voice.language_name().unwrap_or("No language!"));
    }

    println!();
}

println!("Found {} voices", voices.len());
Ok(())
}
```

- For API details, see [DescribeVoices](#) in *AWS SDK for Rust API reference*.

## List Amazon Polly pronunciation lexicons using an AWS SDK

The following code examples show how to list Amazon Polly pronunciation lexicons.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class ListLexicons
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();
        var request = new ListLexiconsRequest();

        try
        {
            Console.WriteLine("All voices: ");

            do
            {
                var response = await client.ListLexiconsAsync(request);
                request.NextToken = response.NextToken;

                response.Lexicons.ForEach(lexicon =>
                {
                    var attributes = lexicon.Attributes;
```

```
        Console.WriteLine($"Name: {lexicon.Name}");
        Console.WriteLine($"\\tAlphabet: {attributes.Alphabet}");
        Console.WriteLine($"\\tLanguageCode:
{attributes.LanguageCode}");
        Console.WriteLine($"\\tLastModified:
{attributes.LastModified}");
        Console.WriteLine($"\\tLexemesCount:
{attributes.LexemesCount}");
        Console.WriteLine($"\\tLexiconArn:
{attributes.LexiconArn}");
        Console.WriteLine($"\\tSize: {attributes.Size}");
    });
}
while (request.NextToken is not null);
}
catch (Exception ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}
}
}
```

- For API details, see [ListLexicons in AWS SDK for .NET API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""
    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
        resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None

    def list_lexicons(self):
        """
        Lists lexicons in the current account.

        :return: The list of lexicons.
        """
        try:
            response = self.polly_client.list_lexicons()
            lexicons = response['Lexicons']
            logger.info("Got %s lexicons.", len(lexicons))
        except ClientError:
            logger.exception("Couldn't get %s.", )
            raise
        else:
            return lexicons
```

- For API details, see [ListLexicons in AWS SDK for Python \(Boto3\) API Reference](#).

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_lexicons(client: &Client) -> Result<(), Error> {
    let resp = client.list_lexicons().send().await?;

    println!("Lexicons:");

    let lexicons = resp.lexicons().unwrap_or_default();

    for lexicon in lexicons {
        println!("  Name:      {}", lexicon.name().unwrap_or_default());
        println!("    Language: {}?\n", lexicon
            .attributes()
            .as_ref()
            .map(|attrib| attrib
                .language_code
                .as_ref()
                .expect("languages must have language codes"))
            .expect("languages must have attributes")
        );
    }

    println!();
    println!("Found {} lexicons.", lexicons.len());
    println!();

    Ok(())
}
```

- For API details, see [ListLexicons in AWS SDK for Rust API reference](#).

## Start an Amazon Polly speech synthesis task using an AWS SDK

The following code example shows how to start an asynchronous Amazon Polly speech synthesis task.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PollyWrapper:
```

```

"""Encapsulates Amazon Polly functions."""
def __init__(self, polly_client, s3_resource):
    """
    :param polly_client: A Boto3 Amazon Polly client.
    :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
    resource.
    """
    self.polly_client = polly_client
    self.s3_resource = s3_resource
    self.voice_metadata = None

    def do_synthesis_task(
        self, text, engine, voice, audio_format, s3_bucket, lang_code=None,
        include_visemes=False, wait_callback=None):
        """
        Start an asynchronous task to synthesize speech or speech marks, wait for
        the task to complete, retrieve the output from Amazon S3, and return the
        data.

        An asynchronous task is required when the text is too long for near-real
        time
        synthesis.

        :param text: The text to synthesize.
        :param engine: The kind of engine used. Can be standard or neural.
        :param voice: The ID of the voice to use.
        :param audio_format: The audio format to return for synthesized speech.
    When
                    speech marks are synthesized, the output format is
    JSON.
                    :param s3_bucket: The name of an existing Amazon S3 bucket that you have
                        write access to. Synthesis output is written to this
                        bucket.
                    :param lang_code: The language code of the voice to use. This has an effect
                        only when a bilingual voice is selected.
                    :param include_visemes: When True, a second request is made to Amazon Polly
                        to synthesize a list of visemes, using the
                        specified
                        text and voice. A viseme represents the visual
                        position
                        of the face and mouth when saying part of a word.
                    :param wait_callback: A callback function that is called periodically
                        during
                        task processing, to give the caller an opportunity to
                        take action, such as to display status.
    :return: The audio stream that contains the synthesized speech and a list
            of visemes that are associated with the speech audio.
    """
    try:
        kwargs = {
            'Engine': engine,
            'OutputFormat': audio_format,
            'OutputS3BucketName': s3_bucket,
            'Text': text,
            'VoiceId': voice}
        if lang_code is not None:
            kwargs['LanguageCode'] = lang_code
        response = self.polly_client.start_speech_synthesis_task(**kwargs)
        speech_task = response['SynthesisTask']
        logger.info("Started speech synthesis task %s.", speech_task['TaskId'])

        viseme_task = None
        if include_visemes:
            kwargs['OutputFormat'] = 'json'
            kwargs['SpeechMarkTypes'] = ['viseme']
            response = self.polly_client.start_speech_synthesis_task(**kwargs)

```

```
        viseme_task = response['SynthesisTask']
        logger.info("Started viseme synthesis task %s.",
viseme_task['TaskId'])
    except ClientError:
        logger.exception("Couldn't start synthesis task.")
        raise
    else:
        bucket = self.s3_resource.Bucket(s3_bucket)
        audio_stream = self._wait_for_task(
            10, speech_task['TaskId'], 'speech', wait_callback, bucket)

        visemes = None
        if include_visemes:
            viseme_data = self._wait_for_task(
                10, viseme_task['TaskId'], 'viseme', wait_callback, bucket)
            visemes = [json.loads(v) for v in
                       viseme_data.read().decode().split() if v]

    return audio_stream, visemes
```

- For API details, see [StartSpeechSynthesisTask](#) in *AWS SDK for Python (Boto3) API Reference*.

## Store an Amazon Polly pronunciation lexicon using an AWS SDK

The following code examples show how to store an Amazon Polly pronunciation lexicon.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class PutLexicon
{
    public static async Task Main()
    {
        string lexiconContent = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +
            "<lexicon version=\"1.0\" xmlns=\"http://www.w3.org/2005/01/
pronunciation-lexicon\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" " +
            "xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-
lexicon http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\" " +
            "alphabet=\"ipa\" xml:lang=\"en-US\">" +
            "<lexeme>test1</lexeme><grapheme>test2</grapheme><alias>test2</alias></lexeme>" +
            "</lexicon>";
        string lexiconName = "SampleLexicon";

        var client = new AmazonPollyClient();
        var putLexiconRequest = new PutLexiconRequest()
        {
            Name = lexiconName,
            Content = lexiconContent,
        };
    }
}
```

```
        try
    {
        var response = await client.PutLexiconAsync(putLexiconRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully created Lexicon:
{lexiconName}.");
        }
        else
        {
            Console.WriteLine($"Could not create Lexicon: {lexiconName}.");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Exception caught: " + ex.Message);
    }
}
```

- For API details, see [PutLexicon](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""
    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
        resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None

    def create_lexicon(self, name, content):
        """
        Creates a lexicon with the specified content. A lexicon contains custom
        pronunciations.

        :param name: The name of the lexicon.
        :param content: The content of the lexicon.
        """
        try:
            self.polly_client.put_lexicon(Name=name, Content=content)
            logger.info("Created lexicon %s.", name)
        except ClientError:
            logger.exception("Couldn't create lexicon %s.")
            raise
```

- For API details, see [PutLexicon](#) in *AWS SDK for Python (Boto3) API Reference*.

Rust

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_lexicon(client: &Client, name: &str, from: &str, to: &str) ->
    Result<(), Error> {
    let content = format!(<?xml version="1.0\" encoding="UTF-8"?>
        <lexicon version="1.0\" xmlns="http://www.w3.org/2005/01/pronunciation-
        lexicon" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon http://
        www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\""
        alphabet="ipa\" xml:lang="en-US\""
        <lexeme><grapheme>{}</grapheme><alias>{}</alias></lexeme>
        </lexicon>", from, to);

    client
        .put_lexicon()
        .name(name)
        .content(content)
        .send()
        .await?;

    println!("Added lexicon");

    Ok(())
}
```

- For API details, see [PutLexicon](#) in *AWS SDK for Rust API reference*.

## Synthesize speech from text with Amazon Polly using an AWS SDK

The following code examples show how to synthesize speech from text with Amazon Polly.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class SynthesizeSpeech
```

```
{
    public static async Task Main()
    {
        string outputFileName = "speech.mp3";
        string text = "Twas brillig, and the slithy toves did gyre and gimbal
in the wabe";

        var client = new AmazonPollyClient();
        var response = await PollySynthesizeSpeech(client, text);

        WriteSpeechToStream(response.AudioStream, outputFileName);
    }

    /// <summary>
    /// Calls the Amazon Polly SynthesizeSpeechAsync method to convert text
    /// to speech.
    /// </summary>
    /// <param name="client">The Amazon Polly client object used to connect
    /// to the Amazon Polly service.</param>
    /// <param name="text">The text to convert to speech.</param>
    /// <returns>A SynthesizeSpeechResponse object that includes an AudioStream
    /// object with the converted text.</returns>
    private static async Task<SynthesizeSpeechResponse>
PollySynthesizeSpeech(IAmazonPolly client, string text)
{
    var synthesizeSpeechRequest = new SynthesizeSpeechRequest()
    {
        OutputFormat = OutputFormat.Mp3,
        VoiceId = VoiceId.Joanna,
        Text = text,
    };

    var synthesizeSpeechResponse =
        await client.SynthesizeSpeechAsync(synthesizeSpeechRequest);

    return synthesizeSpeechResponse;
}

    /// <summary>
    /// Writes the AudioStream returned from the call to
    /// SynthesizeSpeechAsync to a file in MP3 format.
    /// </summary>
    /// <param name="audioStream">The AudioStream returned from the
    /// call to the SynthesizeSpeechAsync method.</param>
    /// <param name="outputFileName">The full path to the file in which to
    /// save the audio stream.</param>
    private static void WriteSpeechToStream(Stream audioStream, string
outputFileName)
{
    var outputStream = new FileStream(
        outputFileName,
        FileMode.Create,
        FileAccess.Write);
    byte[] buffer = new byte[2 * 1024];
    int readBytes;

    while ((readBytes = audioStream.Read(buffer, 0, 2 * 1024)) > 0)
    {
        outputStream.Write(buffer, 0, readBytes);
    }

    // Flushes the buffer to avoid losing the last second or so of
    // the synthesized text.
    outputStream.Flush();
    Console.WriteLine($"Saved {outputFileName} to disk.");
}
}
```

```
}
```

Synthesize speech from text using speech marks with Amazon Polly using an AWS SDK.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class SynthesizeSpeechMarks
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();
        string outputFileName = "speechMarks.json";

        var synthesizeSpeechRequest = new SynthesizeSpeechRequest()
        {
            OutputFormat = OutputFormat.Json,
            SpeechMarkTypes = new List<string>
            {
                SpeechMarkType.Viseme,
                SpeechMarkType.Word,
            },
            VoiceId = VoiceId.Joanna,
            Text = "This is a sample text to be synthesized.",
        };

        try
        {
            using (var outputStream = new FileStream(outputFileName,
FileMode.Create, FileAccess.Write))
            {
                var synthesizeSpeechResponse = await
client.SynthesizeSpeechAsync(synthesizeSpeechRequest);
                var buffer = new byte[2 * 1024];
                int readBytes;

                var inputStream = synthesizeSpeechResponse.AudioStream;
                while ((readBytes = inputStream.Read(buffer, 0, 2 * 1024)) > 0)
                {
                    outputStream.Write(buffer, 0, readBytes);
                }
            }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
    }
}
```

- For API details, see [SynthesizeSpeech](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""
    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3)
        resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None

    def synthesize(
        self, text, engine, voice, audio_format, lang_code=None,
        include_visemes=False):
        """
        Synthesizes speech or speech marks from text, using the specified voice.

        :param text: The text to synthesize.
        :param engine: The kind of engine used. Can be standard or neural.
        :param voice: The ID of the voice to use.
        :param audio_format: The audio format to return for synthesized speech.
    When
                speech marks are synthesized, the output format is
    JSON.
                :param lang_code: The language code of the voice to use. This has an effect
                    only when a bilingual voice is selected.
                :param include_visemes: When True, a second request is made to Amazon Polly
                    to synthesize a list of visemes, using the
    specified
                text and voice. A viseme represents the visual
    position
                of the face and mouth when saying part of a word.
        :return: The audio stream that contains the synthesized speech and a list
            of visemes that are associated with the speech audio.
    """
    try:
        kwargs = {
            'Engine': engine,
            'OutputFormat': audio_format,
            'Text': text,
            'VoiceId': voice}
        if lang_code is not None:
            kwargs['LanguageCode'] = lang_code
        response = self.polly_client.synthesize_speech(**kwargs)
        audio_stream = response['AudioStream']
        logger.info("Got audio stream spoken by %s.", voice)
        visemes = None
        if include_visemes:
            kwargs['OutputFormat'] = 'json'
            kwargs['SpeechMarkTypes'] = ['viseme']
            response = self.polly_client.synthesize_speech(**kwargs)
            visemes = [json.loads(v) for v in
                      response['AudioStream'].read().decode().split() if v]
            logger.info("Got %s visemes.", len(visemes))
    except ClientError:
```

```
        logger.exception("Couldn't get audio stream.")
        raise
    else:
        return audio_stream, visemes
```

- For API details, see [SynthesizeSpeech](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn synthesize(client: &Client, filename: &str) -> Result<(), Error> {
    let content = fs::read_to_string(filename);

    let resp = client
        .synthesize_speech()
        .output_format(OutputFormat::Mp3)
        .text(content.unwrap())
        .voice_id(VoiceId::Joanna)
        .send()
        .await?;

    // Get MP3 data from response and save it
    let mut blob = resp
        .audio_stream
        .collect()
        .await
        .expect("failed to read data");

    let parts: Vec<&str> = filename.split('.').collect();
    let out_file = format!("{}{}", String::from(parts[0]), ".mp3");

    let mut file = tokio::fs::File::create(out_file)
        .await
        .expect("failed to create file");

    file.write_all_buf(&mut blob)
        .await
        .expect("failed to write to file");

    Ok(())
}
```

- For API details, see [SynthesizeSpeech](#) in *AWS SDK for Rust API reference*.

## Scenarios for Amazon Polly using AWS SDKs

The following code examples show how to use Amazon Polly with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Create a lip-sync application with Amazon Polly using an AWS SDK \(p. 1428\)](#)

## Create a lip-sync application with Amazon Polly using an AWS SDK

The following code example shows how to create a lip-sync application with Amazon Polly.

Python

### SDK for Python (Boto3)

Shows how to use Amazon Polly and Tkinter to create a lip-sync application that displays an animated face speaking along with the speech synthesized by Amazon Polly. Lip-sync is accomplished by requesting a list of visemes from Amazon Polly that match up with the synthesized speech.

- Get voice metadata from Amazon Polly and display it in a Tkinter application.
- Get synthesized speech audio and matching viseme speech marks from Amazon Polly.
- Play the audio with synchronized mouth movements in an animated face.
- Submit asynchronous synthesis tasks for long texts and retrieve the output from an Amazon Simple Storage Service (Amazon S3) bucket.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Polly

## Cross-service examples for Amazon Polly using AWS SDKs

The following code examples show how to use Amazon Polly with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Convert text to speech and back to text using an AWS SDK \(p. 1428\)](#)

## Convert text to speech and back to text using an AWS SDK

The following code example shows how to:

- Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file.
- Upload the audio file to an Amazon Simple Storage Service (Amazon S3) bucket.
- Use Amazon Transcribe to convert the audio file to text.
- Display the text.

Rust

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file, upload the audio file to an Amazon Simple Storage Service bucket, use Amazon Transcribe to convert that audio file to text, and display the text.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Polly
- Amazon S3
- Amazon Transcribe

## Code examples for QLDB using AWS SDKs

The following code examples show how to use Amazon QLDB with an AWS software development kit (SDK).

#### Code examples

- [Actions for QLDB using AWS SDKs \(p. 1429\)](#)
  - [Create a QLDB ledger using an AWS SDK \(p. 1429\)](#)
  - [List your QLDB ledgers using an AWS SDK \(p. 1430\)](#)

## Actions for QLDB using AWS SDKs

The following code examples show how to use Amazon QLDB with AWS SDKs. Each example calls an individual service function.

#### Examples

- [Create a QLDB ledger using an AWS SDK \(p. 1429\)](#)
- [List your QLDB ledgers using an AWS SDK \(p. 1430\)](#)

## Create a QLDB ledger using an AWS SDK

The following code example shows how to create a QLDB ledger.

Rust

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_ledger(client: &Client, ledger: &str) -> Result<(), Error> {
    let result = client
        .create_ledger()
        .name(ledger)
        .permissions_mode(PermissionsMode::AllowAll)
        .send()
        .await?;

    println!("ARN: {}", result.arn().unwrap());

    Ok(())
}
```

- For API details, see [CreateLedger](#) in *AWS SDK for Rust API reference*.

## List your QLDB ledgers using an AWS SDK

The following code example shows how to list your QLDB ledgers.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_ledgers(client: &QLDBClient) -> Result<(), Error> {
    let mut pages = client.list_ledgers().into_paginator().page_size(2).send();

    while let Some(page) = pages.next().await {
        println!("* {:?}", page); //Prints an entire page of ledgers.
        for ledger in page.unwrap().ledgers().unwrap() {
            println!("* {:?}", ledger); //Prints the LedgerSummary of a single
        ledger.
        }
    }

    Ok(())
}
```

- For API details, see [ListLedgers](#) in *AWS SDK for Rust API reference*.

## Code examples for Amazon RDS using AWS SDKs

The following code examples show how to use Amazon Relational Database Service with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon RDS using AWS SDKs \(p. 1431\)](#)
  - [Create an Amazon RDS DB instance using an AWS SDK \(p. 1431\)](#)

- [Create an Amazon RDS DB parameter group using an AWS SDK \(p. 1435\)](#)
- [Create a snapshot of an Amazon RDS DB instance using an AWS SDK \(p. 1437\)](#)
- [Delete an Amazon RDS DB instance using an AWS SDK \(p. 1439\)](#)
- [Delete an Amazon RDS DB parameter group using an AWS SDK \(p. 1442\)](#)
- [Describe Amazon RDS DB instances using an AWS SDK \(p. 1444\)](#)
- [Describe Amazon RDS DB parameter groups using an AWS SDK \(p. 1446\)](#)
- [Describe Amazon RDS database engine versions using an AWS SDK \(p. 1448\)](#)
- [Describe options for Amazon RDS DB instances using an AWS SDK \(p. 1451\)](#)
- [Describe parameters in an Amazon RDS DB parameter group using an AWS SDK \(p. 1453\)](#)
- [Describe snapshots of Amazon RDS DB instances using an AWS SDK \(p. 1455\)](#)
- [Modify an Amazon RDS DB instance using an AWS SDK \(p. 1457\)](#)
- [Reboot an Amazon RDS DB instance using an AWS SDK \(p. 1458\)](#)
- [Retrieve attributes that belongs to an Amazon RDS account using an AWS SDK \(p. 1459\)](#)
- [Update parameters in an Amazon RDS DB parameter group using an AWS SDK \(p. 1460\)](#)
- [Scenarios for Amazon RDS using AWS SDKs \(p. 1462\)](#)
  - [Get started with Amazon RDS DB instances using an AWS SDK \(p. 1462\)](#)
- [Cross-service examples for Amazon RDS using AWS SDKs \(p. 1501\)](#)
  - [Create a lending library REST API \(p. 1501\)](#)
  - [Create an Aurora Serverless work item tracker \(p. 1502\)](#)

## Actions for Amazon RDS using AWS SDKs

The following code examples show how to use Amazon Relational Database Service with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create an Amazon RDS DB instance using an AWS SDK \(p. 1431\)](#)
- [Create an Amazon RDS DB parameter group using an AWS SDK \(p. 1435\)](#)
- [Create a snapshot of an Amazon RDS DB instance using an AWS SDK \(p. 1437\)](#)
- [Delete an Amazon RDS DB instance using an AWS SDK \(p. 1439\)](#)
- [Delete an Amazon RDS DB parameter group using an AWS SDK \(p. 1442\)](#)
- [Describe Amazon RDS DB instances using an AWS SDK \(p. 1444\)](#)
- [Describe Amazon RDS DB parameter groups using an AWS SDK \(p. 1446\)](#)
- [Describe Amazon RDS database engine versions using an AWS SDK \(p. 1448\)](#)
- [Describe options for Amazon RDS DB instances using an AWS SDK \(p. 1451\)](#)
- [Describe parameters in an Amazon RDS DB parameter group using an AWS SDK \(p. 1453\)](#)
- [Describe snapshots of Amazon RDS DB instances using an AWS SDK \(p. 1455\)](#)
- [Modify an Amazon RDS DB instance using an AWS SDK \(p. 1457\)](#)
- [Reboot an Amazon RDS DB instance using an AWS SDK \(p. 1458\)](#)
- [Retrieve attributes that belongs to an Amazon RDS account using an AWS SDK \(p. 1459\)](#)
- [Update parameters in an Amazon RDS DB parameter group using an AWS SDK \(p. 1460\)](#)

## Create an Amazon RDS DB instance using an AWS SDK

The following code examples show how to create an Amazon RDS DB instance and wait for it to become available.

## .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create an RDS DB instance with a particular set of properties. Use the
action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbName">Name for the DB instance.</param>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <param name="allocatedStorage">The amount of storage in gibibytes (GiB) to
allocate to the DB instance.</param>
    /// <param name="adminName">Admin user name.</param>
    /// <param name="adminPassword">Admin user password.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
        string parameterGroupName, string dbEngine, string dbEngineVersion,
        string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
{
    var response = await _amazonRDS.CreateDBInstanceAsync(
        new CreateDBInstanceRequest()
    {
        DBName = dbName,
        DBInstanceIdentifier = dbInstanceIdentifier,
        DBParameterGroupName = parameterGroupName,
        Engine = dbEngine,
        EngineVersion = dbEngineVersion,
        DBInstanceClass = instanceClass,
        AllocatedStorage = allocatedStorage,
        MasterUsername = adminName,
        MasterUserPassword = adminPassword
    });
    return response.DBInstance;
}
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createSnapshot(RdsClient rdsClient, String dbInstanceIdentifier, String dbSnapshotIdentifier) {

    try {
        CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.print("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createDatabaseInstance(
    dbInstanceIdentifierVal: String?,
    dbNamedbVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?
) {

    val instanceRequest = CreateDbInstanceRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
        allocatedStorage = 100
        dbName = dbNamedbVal
        engine = "mysql"
        dbInstanceClass = "db.m4.large"
        engineVersion = "8.0.15"
        storageType = "standard"
        masterUsername = masterUsernameVal
        masterUserPassword = masterUserPasswordVal
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceState}")
    }
}

// Waits until the database instance is available.
```

```
suspend fun waitForInstanceReady(dbInstanceIdentifierVal: String?) {
    val sleepTime: Long = 20
    var instanceReady = false
    var instanceReadyStr = ""
    println("Waiting for instance to become available.")

    val instanceRequest = DescribeDbInstancesRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {

                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus.toString()
                    if (instanceReadyStr.contains("available"))
                        instanceReady = true
                    else {
                        println("...$instanceReadyStr")
                        delay(sleepTime * 1000)
                    }
                }
            }
            println("Database instance is available!")
        }
    }
}
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def create_db_instance(
            self, db_name, instance_id, parameter_group_name, db_engine,
            db_engine_version,
            instance_class, storage_type, allocated_storage, admin_name,
            admin_password):
```

```
"""
Creates a DB instance.

:param db_name: The name of the database that is created in the DB
instance.
:param instance_id: The ID to give the newly created DB instance.
:param parameter_group_name: A parameter group to associate with the DB
instance.
:param db_engine: The database engine of a database to create in the DB
instance.
:param db_engine_version: The engine version for the created database.
:param instance_class: The DB instance class for the newly created DB
instance.
:param storage_type: The storage type of the DB instance.
:param allocated_storage: The amount of storage allocated on the DB
instance, in GiBs.
:param admin_name: The name of the admin user for the created database.
:param admin_password: The admin password for the created database.
:returns: Data about the newly created DB instance.
"""

try:
    response = self.rds_client.create_db_instance(
        DBName=db_name,
        DBInstanceIdentifier=instance_id,
        DBParameterGroupName=parameter_group_name,
        Engine=db_engine,
        EngineVersion=db_engine_version,
        DBInstanceClass=instance_class,
        StorageType=storage_type,
        AllocatedStorage=allocated_storage,
        MasterUsername=admin_name,
        MasterUserPassword=admin_password)
    db_inst = response['DBInstance']
except ClientError as err:
    logger.error(
        "Couldn't create DB instance %s. Here's why: %s: %s",
        instance_id,
        err.response['Error']['Code'],
        err.response['Error']['Message'])
    raise
else:
    return db_inst
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an Amazon RDS DB parameter group using an AWS SDK

The following code examples show how to create an Amazon RDS DB parameter group.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create a new DB parameter group. Use the action
DescribeDBParameterGroupsAsync
/// to determine when the DB parameter group is ready to use.
/// </summary>
```

```
/// <param name="name">Name of the DB parameter group.</param>
/// <param name="family">Family of the DB parameter group.</param>
/// <param name="description">Description of the DB parameter group.</param>
/// <returns>The new DB parameter group.</returns>
public async Task<DBParameterGroup> CreateDBParameterGroup(
    string name, string family, string description)
{
    var response = await _amazonRDS.CreateDBParameterGroupAsync(
        new CreateDBParameterGroupRequest()
    {
        DBParameterGroupName = name,
        DBParameterGroupFamily = family,
        Description = description
    });
    return response.DBParameterGroup;
}
```

- For API details, see [CreateDBParameterGroup](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
        .dbParameterGroupName(dbGroupName)
        .dbParameterGroupFamily(dbParameterGroupFamily)
        .description("Created by using the AWS SDK for Java")
        .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is "+
response.dbParameterGroup().dbParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateDBParameterGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def create_parameter_group(self, parameter_group_name, parameter_group_family,
                               description):
        """
        Creates a DB parameter group that is based on the specified parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter group.
        :param parameter_group_family: The family that is used as the basis of the
                                       new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description)
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

- For API details, see [CreateDBParameterGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a snapshot of an Amazon RDS DB instance using an AWS SDK

The following code examples show how to create a snapshot of an Amazon RDS DB instance.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
```

```
/// Create a snapshot of a DB instance.  
/// </summary>  
/// <param name="dbInstanceIdentifier">DB instance identifier.</param>  
/// <param name="snapshotIdentifier">Identifier for the snapshot.</param>  
/// <returns>DB snapshot object.</returns>  
public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier,  
string snapshotIdentifier)  
{  
    var response = await _amazonRDS.CreateDBSnapshotAsync(  
        new CreateDBSnapshotRequest()  
    {  
        DBSnapshotIdentifier = snapshotIdentifier,  
        DBInstanceIdentifier = dbInstanceIdentifier  
    });  
  
    return response.DBSnapshot;  
}
```

- For API details, see [CreateDBSnapshot](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Create an Amazon RDS snapshot.  
public static void createSnapshot(RdsClient rdsClient, String  
dbInstanceIdentifier, String dbSnapshotIdentifier) {  
    try {  
        CreateDbSnapshotRequest snapshotRequest =  
CreateDbSnapshotRequest.builder()  
            .dbInstanceIdentifier(dbInstanceIdentifier)  
            .dbSnapshotIdentifier(dbSnapshotIdentifier)  
            .build();  
  
        CreateDbSnapshotResponse response =  
rdsClient.createDBSnapshot(snapshotRequest);  
        System.out.println("The Snapshot id is " +  
response.dbSnapshot().dbiResourceId());  
  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [CreateDBSnapshot](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:  
    """Encapsulates Amazon RDS DB instance actions."""  
    def __init__(self, rds_client):  
        """  
        :param rds_client: A Boto3 Amazon RDS client.  
        """  
        self.rds_client = rds_client  
  
    @classmethod  
    def from_client(cls):  
        """  
        Instantiates this class from a Boto3 client.  
        """  
        rds_client = boto3.client('rds')  
        return cls(rds_client)  
  
    def create_snapshot(self, snapshot_id, instance_id):  
        """  
        Creates a snapshot of a DB instance.  
  
        :param snapshot_id: The ID to give the created snapshot.  
        :param instance_id: The ID of the DB instance to snapshot.  
        :return: Data about the newly created snapshot.  
        """  
        try:  
            response = self.rds_client.create_db_snapshot(  
                DBSnapshotIdentifier=snapshot_id, DBInstanceIdentifier=instance_id)  
            snapshot = response['DBSnapshot']  
        except ClientError as err:  
            logger.error(  
                "Couldn't create snapshot of %s. Here's why: %s: %s", instance_id,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return snapshot
```

- For API details, see [CreateDBSnapshot](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an Amazon RDS DB instance using an AWS SDK

The following code examples show how to delete an Amazon RDS DB instance.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Delete a particular DB instance.  
/// </summary>  
/// <param name="dbInstanceIdentifier">DB instance identifier.</param>  
/// <returns>DB instance object.</returns>  
public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)  
{  
    var response = await _amazonRDS.DeleteDBInstanceAsync(
```

```
        new DeleteDBInstanceRequest()
    {
        DBInstanceIdentifier = dbInstanceIdentifier,
        SkipFinalSnapshot = true,
        DeleteAutomatedBackups = true
    });

    return response.DBInstance;
}
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteDatabaseInstance( RdsClient rdsClient, String
dbInstanceIdentifier ) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .deleteAutomatedBackups(true)
        .skipFinalSnapshot(true)
        .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.print("The status of the database is " +
response.dbInstance().dbInstanceState());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteDatabaseInstance(dbInstanceIdentifierVal: String?) {
```

```
    val deleteDbInstanceRequest = DeleteDbInstanceRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
        deleteAutomatedBackups = true
        skipFinalSnapshot = true
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
${response.dbInstance?.dbInstanceState}")
    }
}
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id, SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True)
            db_inst = response['DBInstance']
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id, err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
        else:
            return db_inst
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an Amazon RDS DB parameter group using an AWS SDK

The following code examples show how to delete an Amazon RDS DB parameter group.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Delete a DB parameter group. The group cannot be a default DB parameter
group
/// or be associated with any DB instances.
/// </summary>
/// <param name="name">Name of the DB parameter group.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteDBParameterGroup(string name)
{
    var response = await _amazonRDS.DeleteDBParameterGroupAsync(
        new DeleteDBParameterGroupRequest()
    {
        DBParameterGroupName = name,
    });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- For API details, see [DeleteDBParameterGroup](#) in *AWS SDK for .NET API Reference*.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Delete the parameter group after database has been deleted.
// An exception is thrown if you attempt to delete the para group while
database exists.
public static void deleteParaGroup( RdsClient rdsClient, String dbGroupName,
String dbARN) throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN ;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            isDataDel = false ;
        }
    }
}
```

```
        didFind = false;
        int index = 1;
        for (DBInstance instance: instanceList) {
            instanceARN = instance.dbInstanceArn();
            if (instanceARN.compareTo(dbARN) == 0) {
                System.out.println(dbARN + " still exists");
                didFind = true ;
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the
                database ARN.
                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }

    // Delete the para group.
    DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
    .dbParameterGroupName(dbGroupName)
    .build();

    rdsClient.deleteDBParameterGroup(parameterGroupRequest);
    System.out.println(dbGroupName +" was deleted.");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- For API details, see [DeleteDBParameterGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def delete_parameter_group(self, parameter_group_name):
        """
```

```
Deletes a DB parameter group.

:param parameter_group_name: The name of the parameter group to delete.
:return: Data about the parameter group.
"""
try:
    self.rds_client.delete_db_parameter_group(
        DBParameterGroupName=parameter_group_name)
except ClientError as err:
    logger.error(
        "Couldn't delete parameter group %s. Here's why: %s: %s",
        parameter_group_name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
```

- For API details, see [DeleteDBParameterGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe Amazon RDS DB instances using an AWS SDK

The following code examples show how to describe Amazon RDS DB instances.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Returns a list of DB instances.
/// </summary>
/// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
/// <returns>List of DB instances.</returns>
public async Task<List<DBInstance>> DescribeDBInstances(string
dbInstanceIdentifier = null)
{
    var results = new List<DBInstance>();
    var instancesPaginator = _amazonRDS.Paginator.DescribeDBInstances(
        new DescribeDBInstancesRequest
    {
        DBInstanceIdentifier = dbInstanceIdentifier
    });
    // Get the entire list using the paginator.
    await foreach (var instances in instancesPaginator.DBInstances)
    {
        results.Add(instances);
    }
    return results;
}
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeInstances(RdsClient rdsClient) {  
  
    try {  
        DescribeDBInstancesResponse response = rdsClient.describeDBInstances();  
        List<DBInstance> instanceList = response.dbInstances();  
        for (DBInstance instance: instanceList) {  
            System.out.println("Instance ARN is: " + instance.dbInstanceArn());  
            System.out.println("The Engine is " + instance.engine());  
            System.out.println("Connection endpoint is"  
+ instance.endpoint().address());  
        }  
  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeInstances() {  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val response = rdsClient.describeDbInstances(DescribeDbInstancesRequest {})  
        response.dbInstances?.forEach { instance ->  
            println("Instance Identifier is ${instance.dbInstanceIdentifier}")  
            println("The Engine is ${instance.engine}")  
            println("Connection endpoint is ${instance.endpoint?.address}")  
        }  
    }  
}
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:  
    """Encapsulates Amazon RDS DB instance actions."""  
    def __init__(self, rds_client):  
        """  
        :param rds_client: A Boto3 Amazon RDS client.  
        """  
        self.rds_client = rds_client  
  
    @classmethod  
    def from_client(cls):  
        """  
        Instantiates this class from a Boto3 client.  
        """  
        rds_client = boto3.client('rds')  
        return cls(rds_client)  
  
    def get_db_instance(self, instance_id):  
        """  
        Gets data about a DB instance.  
  
        :param instance_id: The ID of the DB instance to retrieve.  
        :return: The retrieved DB instance.  
        """  
        try:  
            response = self.rds_client.describe_db_instances(  
                DBInstanceIdentifier=instance_id)  
            db_inst = response['DBInstances'][0]  
        except ClientError as err:  
            if err.response['Error']['Code'] == 'DBInstanceNotFound':  
                logger.info("Instance %s does not exist.", instance_id)  
            else:  
                logger.error(  
                    "Couldn't get DB instance %s. Here's why: %s: %s", instance_id,  
                    err.response['Error']['Code'], err.response['Error']['Message'])  
                raise  
        else:  
            return db_inst
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe Amazon RDS DB parameter groups using an AWS SDK

The following code examples show how to describe Amazon RDS DB parameter groups.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
    /// <param name="name">Optional name of the DB parameter group to describe.</param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string name = null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
            {
                DBParameterGroupName = name
            });
        return response.DBParameterGroups;
    }
```

- For API details, see [DescribeDBParameterGroups](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDbParameterGroups(RdsClient rdsClient, String dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
            DescribeDbParameterGroupsRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .maxRecords(20)
                .build();

        DescribeDbParameterGroupsResponse response =
            rdsClient.describeDBParameterGroups(groupsRequest);
        List<DBParameterGroup> groups = response.dbParameterGroups();
        for (DBParameterGroup group: groups) {
            System.out.println("The group name is " + group.dbParameterGroupName());
            System.out.println("The group description is " + group.description());
        }
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBParameterGroups](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name)
            parameter_group = response['DBParameterGroups'][0]
        except ClientError as err:
            if err.response['Error']['Code'] == 'DBParameterGroupNotFound':
                logger.info("Parameter group %s does not exist.", parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response['Error']['Code'], err.response['Error']['Message'])
                raise
        else:
            return parameter_group
```

- For API details, see [DescribeDBParameterGroups](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe Amazon RDS database engine versions using an AWS SDK

The following code examples show how to describe Amazon RDS database engine versions.

.NET

**AWS SDK for .NET**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get a list of DB engine versions for a particular DB engine.
/// </summary>
/// <param name="engine">Name of the engine.</param>
/// <param name="dbParameterGroupFamily">Optional parameter group family
name.</param>
/// <returns>List of DBEngineVersions.</returns>
public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string
engine,
    string dbParameterGroupFamily = null)
{
    var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
        new DescribeDBEngineVersionsRequest()
    {
        Engine = engine,
        DBParameterGroupFamily = dbParameterGroupFamily
    });
    return response.DBEngineVersions;
}
```

- For API details, see [DescribeDBEngineVersions](#) in *AWS SDK for .NET API Reference*.

Java

**SDK for Java 2.x**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDBEngines( RdsClient rdsClient ) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .defaultOnly(true)
    .engine("mysql")
    .maxRecords(20)
    .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engine0b: engines) {
            System.out.println("The name of the DB parameter group family for
the database engine is "+engine0b.dbParameterGroupFamily());
            System.out.println("The name of the database engine
"+engine0b.engine());
            System.out.println("The version number of the database engine
"+engine0b.engineVersion());
```

```
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBEngineVersions](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned list
        of engine versions to those that are compatible
        with this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {'Engine': engine}
            if parameter_group_family is not None:
                kwargs['DBParameterGroupFamily'] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response['DBEngineVersions']
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
        else:
            return versions
```

- For API details, see [DescribeDBEngineVersions](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe options for Amazon RDS DB instances using an AWS SDK

The following code examples show how to describe options for Amazon RDS DB instances.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Get a list of orderable DB instance options for a specific
/// engine and engine version.
///</summary>
///<param name="engine">Name of the engine.</param>
///<param name="engineVersion">Version of the engine.</param>
///<returns>List of OrderableDBInstanceOptions.</returns>
public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
{
    // Use a paginator to get a list of DB instance options.
    var results = new List<OrderableDBInstanceOption>();
    var paginateInstanceOptions =
_amazonRDS.Paginator.DescribeOrderableDBInstanceOptions(
    new DescribeOrderableDBInstanceOptionsRequest()
    {
        Engine = engine,
        EngineVersion = engineVersion,
    });
    // Get the entire list using the paginator.
    await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
    {
        results.Add(instanceOptions);
    }
    return results;
}
```

- For API details, see [DescribeOrderableDBInstanceOptions](#) in *AWS SDK for .NET API Reference*.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get a list of allowed engine versions.
public static void getAvailableEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
```

```
try {
    DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .dbParameterGroupFamily(dbParameterGroupFamily)
    .engine("mysql")
    .build();

    DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
    List<DBEngineVersion> dbEngines = response.dbEngineVersions();
    for (DBEngineVersion dbEngine: dbEngines) {
        System.out.println("The engine version is "
+dbEngine.engineVersion());
        System.out.println("The engine description is "
+dbEngine.dbEngineDescription());
    }

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- For API details, see [DescribeOrderableDBInstancesOptions](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
        instance.
        :param db_engine_version: The engine version that must be supported by the
        DB instance.
        :return: The list of DB instance options that can be used to create a
        compatible DB instance.
        """

```

```
try:
    inst_opts = []
    paginator =
self.rds_client.get_paginator('describe_orderable_db_instance_options')
    for page in paginator.paginate(Engine=db_engine,
    EngineVersion=db_engine_version):
        inst_opts += page['OrderableDBInstanceOptions']
except ClientError as err:
    logger.error(
        "Couldn't get orderable DB instances. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return inst_opts
```

- For API details, see [DescribeOrderableDBInstanceOptions](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe parameters in an Amazon RDS DB parameter group using an AWS SDK

The following code examples show how to describe parameters in an Amazon RDS DB parameter group.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Get a list of DB parameters from a specific parameter group.
///</summary>
///<param name="dbParameterGroupName">Name of a specific DB parameter group.</param>
///<param name="source">Optional source for selecting parameters.</param>
///<returns>List of parameter values.</returns>
public async Task<List<Parameter>> DescribeDBParameters(string
dbParameterGroupName, string source = null)
{
    var results = new List<Parameter>();
    var paginateParameters = _amazonRDS.Paginator.DescribeDBParameters(
        new DescribeDBParametersRequest()
    {
        DBParameterGroupName = dbParameterGroupName,
        Source = source
    });
    // Get the entire list using the paginator.
    await foreach (var parameters in paginateParameters.Parameters)
    {
        results.Add(parameters);
    }
    return results;
}
```

- For API details, see [DescribeDBParameters](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .source("user")
                .build();
        }

        DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para: dbParameters) {
            // Only print out information about either auto_increment_offset or
auto_increment_increment.
            paraName = para.parameterName();
            if ( (paraName.compareTo("auto_increment_offset") ==0) ||
(paraName.compareTo("auto_increment_increment ") ==0)) {
                System.out.println("**** The parameter name is " + paraName);
                System.out.println("**** The parameter value is " +
para.parameterValue());
                System.out.println("**** The parameter data type is " +
para.dataType());
                System.out.println("**** The parameter description is " +
para.description());
                System.out.println("**** The parameter allowed values is " +
para.allowedValues());
            }
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBParameters](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_parameters(self, parameter_group_name, name_prefix='', source=None):
        """
        Gets the parameters that are contained in a DB parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
            filtered
                to contain only parameters that start with this prefix.
        :param source: When specified, only parameters from this source are
            retrieved.
                For example, a source of 'user' retrieves only parameters
            that
                were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {'DBParameterGroupName': parameter_group_name}
            if source is not None:
                kwargs['Source'] = source
            parameters = []
            paginator = self.rds_client.getPaginator('describe_db_parameters')
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p for p in page['Parameters'] if
                    p['ParameterName'].startswith(name_prefix)]
            except ClientError as err:
                logger.error(
                    "Couldn't get parameters for %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response['Error']['Code'], err.response['Error']['Message'])
                raise
            else:
                return parameters
        
```

- For API details, see [DescribeDBParameters](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe snapshots of Amazon RDS DB instances using an AWS SDK

The following code examples show how to describe snapshots of Amazon RDS DB instances.

## .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Return a list of DB snapshots for a particular DB instance.
/// </summary>
/// <param name="dbInstanceIdentifier">DB instance identifier.</param>
/// <returns>List of DB snapshots.</returns>
public async Task<List<DBSnapshot>> DescribeDBSnapshots(string
dbInstanceIdentifier)
{
    var results = new List<DBSnapshot>();
    var snapshotsPaginator = _amazonRDS.Paginator.DescribeDBSnapshots(
        new DescribeDBSnapshotsRequest()
    {
        DBInstanceIdentifier = dbInstanceIdentifier
    });

    // Get the entire list using the paginator.
    await foreach (var snapshots in snapshotsPaginator.DBSnapshots)
    {
        results.Add(snapshots);
    }
    return results;
}
```

- For API details, see [DescribeDBSnapshots in AWS SDK for .NET API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)
```

```
def get_snapshot(self, snapshot_id):
    """
    Gets a DB instance snapshot.

    :param snapshot_id: The ID of the snapshot to retrieve.
    :return: The retrieved snapshot.
    """
    try:
        response = self.rds_client.describe_db_snapshots(
            DBSnapshotIdentifier=snapshot_id)
        snapshot = response['DBSnapshots'][0]
    except ClientError as err:
        logger.error(
            "Couldn't get snapshot %s. Here's why: %s: %s",
            snapshot_id,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return snapshot
```

- For API details, see [DescribeDBSnapshots in AWS SDK for Python \(Boto3\) API Reference](#).

## Modify an Amazon RDS DB instance using an AWS SDK

The following code examples show how to modify an Amazon RDS DB instance.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void updateInstance(RdsClient rdsClient, String
dbInstanceIdentifier, String masterUserPassword) {

    try {
        // For a demo - modify the DB instance by modifying the master
        // password.
        ModifyDbInstanceRequest modifyDbInstanceRequest =
        ModifyDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .publiclyAccessible(true)
            .masterUserPassword(masterUserPassword)
            .build();

        ModifyDbInstanceResponse instanceResponse =
        rdsClient.modifyDBInstance(modifyDbInstanceRequest);
        System.out.print("The ARN of the modified database is: "
        +instanceResponse.dbInstance().dbInstanceArn());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [ModifyDBInstance in AWS SDK for Java 2.x API Reference](#).

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun updateInstance(dbInstanceIdentifierVal: String?, masterUserPasswordVal: String?) {  
  
    val request = ModifyDbInstanceRequest {  
        dbInstanceIdentifier = dbInstanceIdentifierVal  
        publiclyAccessible = true  
        masterUserPassword = masterUserPasswordVal  
    }  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val instanceResponse = rdsClient.modifyDbInstance(request)  
        println("The ARN of the modified database is  
        ${instanceResponse.dbInstance?.dbInstanceArn}")  
    }  
}
```

- For API details, see [ModifyDBInstance](#) in *AWS SDK for Kotlin API reference*.

## Reboot an Amazon RDS DB instance using an AWS SDK

The following code example shows how to reboot an Amazon RDS DB instance.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void rebootInstance(RdsClient rdsClient, String  
dbInstanceIdentifier ) {  
  
    try {  
        RebootDbInstanceRequest rebootDbInstanceRequest =  
RebootDbInstanceRequest.builder()  
            .dbInstanceIdentifier(dbInstanceIdentifier)  
            .build();  
  
        RebootDbInstanceResponse instanceResponse =  
rdsClient.rebootDBInstance(rebootDbInstanceRequest);  
        System.out.print("The database "+  
instanceResponse.dbInstance().dbInstanceArn() +" was rebooted");  
  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

```
}
```

- For API details, see [RebootDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

## Retrieve attributes that belongs to an Amazon RDS account using an AWS SDK

The following code examples show how to retrieve attributes that belong to an Amazon RDS account.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getAccountAttributes(RdsClient rdsClient) {  
  
    try {  
        DescribeAccountAttributesResponse response =  
rdsClient.describeAccountAttributes();  
        List<AccountQuota> quotasList = response.accountQuotas();  
        for (AccountQuota quotas: quotasList) {  
            System.out.println("Name is: "+quotas.accountQuotaName());  
            System.out.println("Max value is " +quotas.max());  
        }  
  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DescribeAccountAttributes](#) in *AWS SDK for Java 2.x API Reference*.

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getAccountAttributes() {  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val response =  
rdsClient.describeAccountAttributes(DescribeAccountAttributesRequest {})  
        response.accountQuotas?.forEach { quotas ->  
            val response = response.accountQuotas  
            println("Name is: ${quotas.accountQuotaName}")
```

```
        }
    }
}
```

- For API details, see [DescribeAccountAttributes](#) in *AWS SDK for Kotlin API reference*.

## Update parameters in an Amazon RDS DB parameter group using an AWS SDK

The following code examples show how to update parameters in an Amazon RDS DB parameter group.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Update a DB parameter group. Use the action DescribeDBParameterGroupsAsync
/// to determine when the DB parameter group is ready to use.
/// </summary>
/// <param name="name">Name of the DB parameter group.</param>
/// <param name="parameters">List of parameters. Maximum of 20 per request.</param>
public async Task<string> ModifyDBParameterGroup(
    string name, List<Parameter> parameters)
{
    var response = await _amazonRDS.ModifyDBParameterGroupAsync(
        new ModifyDBParameterGroupRequest()
    {
        DBParameterGroupName = name,
        Parameters = parameters,
    });
    return response.DBParameterGroupName;
}
```

- For API details, see [ModifyDBParameterGroup](#) in *AWS SDK for .NET API Reference*.

### Java

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
```

```
.parameterName("auto_increment_offset")
.applyMethod("immediate")
.parameterValue("5")
.build();

List<Parameter> paraList = new ArrayList<>();
paraList.add(parameter1);
ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
.dbParameterGroupName(dbGroupName)
.parameters(paraList)
.build();

ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
System.out.println("The parameter group "+
response.dbParameterGroupName() +" was successfully modified");

} catch (RdsException e) {
System.out.println(e.getLocalizedMessage());
System.exit(1);
}
}
```

- For API details, see [ModifyDBParameterGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
                Parameters=update_parameters)
```

```
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

- For API details, see [ModifyDBParameterGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios for Amazon RDS using AWS SDKs

The following code examples show how to use Amazon Relational Database Service with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Get started with Amazon RDS DB instances using an AWS SDK \(p. 1462\)](#)

## Get started with Amazon RDS DB instances using an AWS SDK

The following code examples show how to:

- Create a custom DB parameter group and set parameter values.
- Create a DB instance that's configured to use the parameter group. The DB instance also contains a database.
- Take a snapshot of the instance.
- Delete the instance and parameter group.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
/// <summary>
/// Scenario for RDS DB instance example.
/// </summary>
public class RDSDatabaseScenario
{
    /*
     Before running this .NET code example, set up your development environment,
     including your credentials.

     This .NET example performs the following tasks:
     1. Returns a list of the available DB engine families using the
        DescribeDBEngineVersionsAsync method.
     2. Selects an engine family and creates a custom DB parameter group using the
        CreateDBParameterGroupAsync method.
     3. Gets the parameter groups using the DescribeDBParameterGroupsAsync method.
```

```
    4. Gets parameters in the group using the DescribeDBParameters method.
    5. Parses and displays parameters in the group.
    6. Modifies both the auto_increment_offset and auto_increment_increment
parameters
        using the ModifyDBParameterGroupAsync method.
    7. Gets and displays the updated parameters using the DescribeDBParameters
method with a source of "user".
    8. Gets a list of allowed engine versions using the
DescribeDBEngineVersionsAsync method.
    9. Displays and selects from a list of micro instance classes available for
the selected engine and version.
    10. Creates an RDS DB instance that contains a MySql database and uses the
parameter group
        using the CreateDBInstanceAsync method.
    11. Waits for DB instance to be ready using the DescribeDBInstancesAsync
method.
    12. Prints out the connection endpoint string for the new DB instance.
    13. Creates a snapshot of the DB instance using the CreateDBSnapshotAsync
method.
    14. Waits for DB snapshot to be ready using the DescribeDBSnapshots method.
    15. Deletes the DB instance using the DeleteDBInstanceAsync method.
    16. Waits for DB instance to be deleted using the DescribeDbInstances method.
    17. Deletes the parameter group using the DeleteDBParameterGroupAsync.
*/
private static readonly string sepBar = new('-', 80);
private static RDSWrapper rdsWrapper = null!;
private static ILogger logger = null!;
private static readonly string engine = "mysql";
static async Task Main(string[] args)
{
    // Set up dependency injection for the Amazon RDS service.
    using var host = Host.CreateDefaultBuilder(args)
        .ConfigureLogging(logging =>
            logging.AddFilter("System", LogLevel.Debug)
                .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
        .ConfigureServices(_>, services =>
            services.AddAWSService<IAmazonRDS>()
                .AddTransient<RDSWrapper>()
        )
        .Build();

    logger = LoggerFactory.Create(builder =>
    {
        builder.AddConsole();
    }).CreateLogger<RDSInstanceScenario>();

    rdsWrapper = host.Services.GetRequiredService<RDSWrapper>();

    Console.WriteLine(sepBar);
    Console.WriteLine(
        "Welcome to the Amazon Relational Database Service (Amazon RDS) DB
instance scenario example.");
    Console.WriteLine(sepBar);

    try
    {
        var parameterGroupFamily = await ChooseParameterGroupFamily();

        var parameterGroup = await
CreateDbParameterGroup(parameterGroupFamily);

        var parameters = await
DescribeParametersInGroup(parameterGroup.DBParameterGroupName,
```

```
        new List<string> { "auto_increment_offset",
"auto_increment_increment" });

        await ModifyParameters(parameterGroup.DBParameterGroupName,
parameters);

        await
DescribeUserSourceParameters(parameterGroup.DBParameterGroupName);

        var engineVersionChoice = await
ChooseDbEngineVersion(parameterGroupFamily);

        var instanceChoice = await ChooseDbInstanceClass(engine,
engineVersionChoice.EngineVersion);

        var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;

        var newInstance = await CreateRdsNewInstance(parameterGroup, engine,
engineVersionChoice.EngineVersion,
            instanceChoice.DBInstanceClass, newInstanceIdentifier);
        if (newInstance != null)
        {
            DisplayConnectionString(newInstance);

            await CreateSnapshot(newInstance);

            await DeleteRdsInstance(newInstance);
        }

        await DeleteParameterGroup(parameterGroup);

        Console.WriteLine("Scenario complete.");
        Console.WriteLine(sepBar);
    }
    catch (Exception ex)
    {
        logger.LogError(ex, "There was a problem executing the scenario.");
    }
}

/// <summary>
/// Choose the RDS DB parameter group family from a list of available options.
/// </summary>
/// <returns>The selected parameter group family.</returns>
public static async Task<string> ChooseParameterGroupFamily()
{
    Console.WriteLine(sepBar);
    // 1. Get a list of available engines.
    var engines = await rdsWrapper.DescribeDBEngineVersions(engine);

    Console.WriteLine("1. The following is a list of available DB parameter
group families:");
    int i = 1;
    var parameterGroupFamilies = engines.GroupBy(e =>
e.DBParameterGroupFamily).ToList();
    foreach (var parameterGroupFamily in parameterGroupFamilies)
    {
        // List the available parameter group families.
        Console.WriteLine(
            $"{i}. Family: {parameterGroupFamily.Key}");
        i++;
    }

    var choiceNumber = 0;
    while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
{
```

```
Console.WriteLine("Select an available DB parameter group family by
entering a number from the list above:");
var choice = Console.ReadLine();
Int32.TryParse(choice, out choiceNumber);
}
var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber - 1];
Console.WriteLine(sepBar);
return parameterGroupFamilyChoice.Key;
}

/// <summary>
/// Create and get information on a DB parameter group.
/// </summary>
/// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the new
DB parameter group.</param>
/// <returns>The new DBParameterGroup.</returns>
public static async Task<DBParameterGroup> CreateDbParameterGroup(string
dbParameterGroupFamily)
{
    Console.WriteLine(sepBar);
    Console.WriteLine($"2. Create new DB parameter group with family
{dbParameterGroupFamily}:");

    var parameterGroup = await rdsWrapper.CreateDBParameterGroup(
        "ExampleParameterGroup-" + DateTime.Now.Ticks,
        dbParameterGroupFamily, "New example parameter group");

    var groupInfo =
        await rdsWrapper.DescribeDBParameterGroups(parameterGroup
            .DBParameterGroupName);

    Console.WriteLine(
        $"3. New DB parameter group: \n\t{groupInfo[0].Description}, \n\tARN
{groupInfo[0].DBParameterGroupArn}");
    Console.WriteLine(sepBar);
    return parameterGroup;
}

/// <summary>
/// Get and describe parameters from a DBParameterGroup.
/// </summary>
/// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
/// <param name="parameterNames">Optional specific names of parameters to
describe.</param>
/// <returns>The list of requested parameters.</returns>
public static async Task<List<Parameter>> DescribeParametersInGroup(string
parameterGroupName, List<string>? parameterNames = null)
{
    Console.WriteLine(sepBar);
    Console.WriteLine("4. Get some parameters from the group.");
    Console.WriteLine(sepBar);

    var parameters =
        await rdsWrapper.DescribeDBParameters(parameterGroupName);

    var matchingParameters =
        parameters.Where(p => parameterNames == null ||

parameterNames.Contains(p.ParameterName)).ToList();

    Console.WriteLine("5. Parameter information:");
    matchingParameters.ForEach(p =>
        Console.WriteLine(
            $"\n\tParameter: {p.ParameterName}." +
            $" \n\tDescription: {p.Description}." +
            $" \n\tAllowed Values: {p.AllowedValues}." +
            $" \n\tValue: {pParameterValue}."));
}
```

```
        Console.WriteLine(sepBar);

        return matchingParameters;
    }

    ///<summary>
    ///<summary>
    ///</summary>
    ///<param name="parameterGroupName">Name of the DBParameterGroup.</param>
    ///<param name="parameters">The parameters to modify.</param>
    ///<returns>Async task.</returns>
    public static async Task ModifyParameters(string parameterGroupName,
List<Parameter> parameters)
{
    Console.WriteLine(sepBar);
    Console.WriteLine("6. Modify some parameters in the group.");

    foreach (var p in parameters)
    {
        if (p.IsModifiable && p.DataType == "integer")
        {
            int newValue = 0;
            while (newValue == 0)
            {
                Console.WriteLine(
                    $"Enter a new value for {p.ParameterName} from the allowed
values {p.AllowedValues} ");

                var choice = Console.ReadLine();
                Int32.TryParse(choice, out newValue);
            }
            pParameterValue = newValue.ToString();
        }
    }

    await rdsWrapper.ModifyDBParameterGroup(parameterGroupName, parameters);

    Console.WriteLine(sepBar);
}

    ///<summary>
    ///<summary>
    ///</summary>
    ///<param name="parameterGroupName">Name of the DBParameterGroup.</param>
    ///<returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string
parameterGroupName)
{
    Console.WriteLine(sepBar);
    Console.WriteLine("7. Describe user source parameters in the group.");

    var parameters =
        await rdsWrapper.DescribeDBParameters(parameterGroupName, "user");

    parameters.ForEach(p =>
        Console.WriteLine(
            $"\\n\\tParameter: {p.ParameterName}." +
            $"\\n\\tDescription: {p.Description}." +
            $"\\n\\tAllowed Values: {p.AllowedValues}." +
            $"\\n\\tValue: {pParameterValue}"));

    Console.WriteLine(sepBar);
}
```

```
    ///<summary>
    /// Choose a DB engine version.
    ///</summary>
    ///<param name="dbParameterGroupFamily">DB parameter group family for engine choice.</param>
    ///<returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDbEngineVersion(string dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await rdsWrapper.DescribeDBEngineVersions(engine,
        dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group family {dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"{i}. Engine: {version.Engine} Version {version.EngineVersion}.");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    ///<summary>
    /// Choose a DB instance class for a particular engine and engine version.
    ///</summary>
    ///<param name="engine">DB engine for DB instance choice.</param>
    ///<param name="engineVersion">DB engine version for DB instance choice.</param>
    ///<returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption>
ChooseDbInstanceClass(string engine, string engineVersion)
{
    Console.WriteLine(sepBar);
    // Get a list of allowed DB instance classes.
    var allowedInstances =
        await rdsWrapper.DescribeOrderableDBInstanceOptions(engine,
    engineVersion);

    Console.WriteLine($"8. Available micro DB instance classes for engine {engine} and version {engineVersion}:");
    int i = 1;

    // Filter to micro instances for this example.
    allowedInstances = allowedInstances
        .Where(i => i.DBInstanceClass.Contains("micro")).ToList();
```

```
foreach (var instance in allowedInstances)
{
    Console.WriteLine(
        $"\\t{i}. Instance class: {instance.DBInstanceClass} (storage type
{instance.StorageType})");
    i++;
}

var choiceNumber = 0;
while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
{
    Console.WriteLine("9. Select an available DB instance class by entering
a number from the list above:");
    var choice = Console.ReadLine();
    Int32.TryParse(choice, out choiceNumber);
}

var instanceChoice = allowedInstances[choiceNumber - 1];
Console.WriteLine(sepBar);
return instanceChoice;
}

///<summary>
/// Create a new RDS DB instance.
///</summary>
///<param name="parameterGroup">Parameter group to use for the DB instance.</
param>
///<param name="engineName">Engine to use for the DB instance.</param>
///<param name="engineVersion">Engine version to use for the DB instance.</
param>
///<param name="instanceClass">Instance class to use for the DB instance.</
param>
///<param name="instanceIdentifier">Instance identifier to use for the DB
instance.</param>
///<returns>The new DB instance.</returns>
public static async Task<DBInstance?> CreateRdsNewInstance(DBParameterGroup
parameterGroup,
    string engineName, string engineVersion, string instanceClass, string
instanceIdentifier)
{
    Console.WriteLine(sepBar);
    Console.WriteLine($"10. Create a new DB instance with identifier
{instanceIdentifier}.");
    bool isInstanceReady = false;
    DBInstance newInstance;
    var instances = await rdsWrapper.DescribeDBInstances();
    isInstanceReady = instances.FirstOrDefault(i =>
        i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
"available";

    if (isInstanceReady)
    {
        Console.WriteLine("Instance already created.");
        newInstance = instances.First(i => i.DBInstanceIdentifier ==
instanceIdentifier);
    }
    else
    {
        Console.WriteLine("Please enter an admin user name:");
        var username = Console.ReadLine();

        Console.WriteLine("Please enter an admin password:");
        var password = Console.ReadLine();

        newInstance = await rdsWrapper.CreateDBInstance(
            "ExampleInstance",
```

```
        instanceIdentifier,
        parameterGroup.DBParameterGroupName,
        engineName,
        engineVersion,
        instanceClass,
        20,
        username,
        password
    );
}

// 11. Wait for the DB instance to be ready.

Console.WriteLine("11. Waiting for DB instance to be ready...");
while (!isInstanceReady)
{
    instances = await
rdsWrapper.DescribeDBInstances(instanceIdentifier);
    isInstanceReady = instances.FirstOrDefault()?.DBInstanceState ==
"available";
    newInstance = instances.First();
    Thread.Sleep(30000);
}
}

Console.WriteLine(sepBar);
return newInstance;
}

/// <summary>
/// Display a connection string for an RDS DB instance.
/// </summary>
/// <param name="instance">The DB instance to use to get a connection string.</param>
public static void DisplayConnectionString(DBInstance instance)
{
    Console.WriteLine(sepBar);
    // Display the connection string.
    Console.WriteLine("12. New DB instance connection string: ");
    Console.WriteLine(
        $"\\n{engine} -h {instance.Endpoint.Address} -P {instance.Endpoint.Port}"
    +
        $"-u {instance.MasterUsername} -p [YOUR PASSWORD]\\n");
    Console.WriteLine(sepBar);
}

/// <summary>
/// Create a snapshot from an RDS DB instance.
/// </summary>
/// <param name="instance">DB instance to use when creating a snapshot.</param>
/// <returns>The snapshot object.</returns>
public static async Task<DBSnapshot> CreateSnapshot(DBInstance instance)
{
    Console.WriteLine(sepBar);
    // Create a snapshot.
    Console.WriteLine($"13. Creating snapshot from DB instance
{instance.DBInstanceIdentifier}.");
    var snapshot = await
rdsWrapper.CreateDBSnapshot(instance.DBInstanceIdentifier, "ExampleSnapshot-" +
DateTime.Now.Ticks);

    // Wait for the snapshot to be available
    bool isSnapshotReady = false;

    Console.WriteLine($"14. Waiting for snapshot to be ready...");
    while (!isSnapshotReady)
```

```
{  
    var snapshots = await  
rdsWrapper.DescribeDBSnapshots(instance.DBInstanceIdentifier);  
    isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";  
    snapshot = snapshots.First();  
    Thread.Sleep(30000);  
}  
  
Console.WriteLine(  
    $"Snapshot {snapshot.DBSnapshotIdentifier} status is  
{snapshot.Status}.");  
    Console.WriteLine(sepBar);  
    return snapshot;  
}  
  
/// <summary>  
/// Delete an RDS DB instance.  
/// </summary>  
/// <param name="instance">The DB instance to delete.</param>  
/// <returns>Async task.</returns>  
public static async Task DeleteRdsInstance(DBInstance newInstance)  
{  
    Console.WriteLine(sepBar);  
    // Delete the DB instance.  
    Console.WriteLine($"15. Delete the DB instance  
{newInstance.DBInstanceIdentifier}.");  
    await rdsWrapper.DeleteDBInstance(newInstance.DBInstanceIdentifier);  
  
    // Wait for the DB instance to delete.  
    Console.WriteLine($"16. Waiting for the DB instance to delete...");  
    bool isInstanceDeleted = false;  
  
    while (!isInstanceDeleted)  
    {  
        var instance = await rdsWrapper.DescribeDBInstances();  
        isInstanceDeleted = instance.All(i => i.DBInstanceIdentifier !=  
newInstance.DBInstanceIdentifier);  
        Thread.Sleep(30000);  
    }  
  
    Console.WriteLine("DB instance deleted.");  
    Console.WriteLine(sepBar);  
}  
  
/// <summary>  
/// Delete a DB parameter group.  
/// </summary>  
/// <param name="parameterGroup">The parameter group to delete.</param>  
/// <returns>Async task.</returns>  
public static async Task DeleteParameterGroup(DBParameterGroup parameterGroup)  
{  
    Console.WriteLine(sepBar);  
    // Delete the parameter group.  
    Console.WriteLine($"17. Delete the DB parameter group  
{parameterGroup.DBParameterGroupName}.");  
    await  
rdsWrapper.DeleteDBParameterGroup(parameterGroup.DBParameterGroupName);  
  
    Console.WriteLine(sepBar);  
}
```

---

Wrapper methods used by the scenario for DB instance actions.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with DB
instance operations.
/// </summary>
public partial class RDSSwrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public RDSSwrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }

    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="dbParameterGroupFamily">Optional parameter group family
name.</param>
    /// <returns>List of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string
engine,
        string dbParameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = dbParameterGroupFamily
            });
        return response.DBEngineVersions;
    }

    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginator.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }

    /// <summary>
```

```

    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstances(string
dbInstanceIdentifier = null)
{
    var results = new List<DBInstance>();
    var instancesPaginator = _amazonRDS.Paginator.DescribeDBInstances(
        new DescribeDBInstancesRequest
    {
        DBInstanceIdentifier = dbInstanceIdentifier
    });
    // Get the entire list using the paginator.
    await foreach (var instances in instancesPaginator.DBInstances)
    {
        results.Add(instances);
    }
    return results;
}

    /// <summary>
    /// Create an RDS DB instance with a particular set of properties. Use the
action DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbName">Name for the DB instance.</param>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <param name="allocatedStorage">The amount of storage in gibibytes (GiB) to
allocate to the DB instance.</param>
    /// <param name="adminName">Admin user name.</param>
    /// <param name="adminPassword">Admin user password.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
        string parameterGroupName, string dbEngine, string dbEngineVersion,
        string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
{
    var response = await _amazonRDS.CreateDBInstanceAsync(
        new CreateDBInstanceRequest()
    {
        DBName = dbName,
        DBInstanceIdentifier = dbInstanceIdentifier,
        DBParameterGroupName = parameterGroupName,
        Engine = dbEngine,
        EngineVersion = dbEngineVersion,
        DBInstanceClass = instanceClass,
        AllocatedStorage = allocatedStorage,
        MasterUsername = adminName,
        MasterUserPassword = adminPassword
    });
    return response.DBInstance;
}

```

```
/// <summary>
/// Delete a particular DB instance.
/// </summary>
/// <param name="dbInstanceIdentifier">DB instance identifier.</param>
/// <returns>DB instance object.</returns>
public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
{
    var response = await _amazonRDS.DeleteDBInstanceAsync(
        new DeleteDBInstanceRequest()
    {
        DBInstanceIdentifier = dbInstanceIdentifier,
        SkipFinalSnapshot = true,
        DeleteAutomatedBackups = true
    });

    return response.DBInstance;
}
```

Wrapper methods used by the scenario for DB parameter groups.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
/// parameter groups.
/// </summary>
public partial class RDSWrapper
{

    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
    /// <param name="name">Optional name of the DB parameter group to describe.</param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string name =
        null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
        {
            DBParameterGroupName = name
        });
        return response.DBParameterGroups;
    }

    /// <summary>
    /// Create a new DB parameter group. Use the action
    /// DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="name">Name of the DB parameter group.</param>
    /// <param name="family">Family of the DB parameter group.</param>
    /// <param name="description">Description of the DB parameter group.</param>
    /// <returns>The new DB parameter group.</returns>
    public async Task<DBParameterGroup> CreateDBParameterGroup(
        string name, string family, string description)
    {
        var response = await _amazonRDS.CreateDBParameterGroupAsync(
            new CreateDBParameterGroupRequest()
        {
            DBParameterGroupName = name,
```

```
        DBParameterGroupFamily = family,
        Description = description
    });
    return response.DBParameterGroup;
}

/// <summary>
/// Update a DB parameter group. Use the action DescribeDBParameterGroupsAsync
/// to determine when the DB parameter group is ready to use.
/// </summary>
/// <param name="name">Name of the DB parameter group.</param>
/// <param name="parameters">List of parameters. Maximum of 20 per request.</param>
public async Task<string> ModifyDBParameterGroup(
    string name, List<Parameter> parameters)
{
    var response = await _amazonRDS.ModifyDBParameterGroupAsync(
        new ModifyDBParameterGroupRequest()
    {
        DBParameterGroupName = name,
        Parameters = parameters,
    });
    return response.DBParameterGroupName;
}

/// <summary>
/// Delete a DB parameter group. The group cannot be a default DB parameter
group
/// or be associated with any DB instances.
/// </summary>
/// <param name="name">Name of the DB parameter group.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteDBParameterGroup(string name)
{
    var response = await _amazonRDS.DeleteDBParameterGroupAsync(
        new DeleteDBParameterGroupRequest()
    {
        DBParameterGroupName = name,
    });
    return response.HttpStatusCode == HttpStatusCode.OK;
}

/// <summary>
/// Get a list of DB parameters from a specific parameter group.
/// </summary>
/// <param name="dbParameterGroupName">Name of a specific DB parameter group.</param>
/// <param name="source">Optional source for selecting parameters.</param>
/// <returns>List of parameter values.</returns>
public async Task<List<Parameter>> DescribeDBParameters(string
dbParameterGroupName, string source = null)
{
    var results = new List<Parameter>();
    var paginateParameters = _amazonRDS.Paginator.DescribeDBParameters(
        new DescribeDBParametersRequest()
    {
        DBParameterGroupName = dbParameterGroupName,
        Source = source
    });
}
```

```
// Get the entire list using the paginator.  
await foreach (var parameters in paginateParameters.Parameters)  
{  
    results.Add(parameters);  
}  
return results;  
}
```

Wrapper methods used by the scenario for DB snapshot actions.

```
/// <summary>  
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with  
/// snapshots.  
/// </summary>  
public partial class RDSWrapper  
{  
  
    /// <summary>  
    /// Create a snapshot of a DB instance.  
    /// </summary>  
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>  
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>  
    /// <returns>DB snapshot object.</returns>  
    public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier,  
string snapshotIdentifier)  
    {  
        var response = await _amazonRDS.CreateDBSnapshotAsync(  
            new CreateDBSnapshotRequest()  
            {  
                DBSnapshotIdentifier = snapshotIdentifier,  
                DBInstanceIdentifier = dbInstanceIdentifier  
            });  
  
        return response.DBSnapshot;  
    }  
  
    /// <summary>  
    /// Return a list of DB snapshots for a particular DB instance.  
    /// </summary>  
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>  
    /// <returns>List of DB snapshots.</returns>  
    public async Task<List<DBSnapshot>> DescribeDBSnapshots(string  
dbInstanceIdentifier)  
    {  
        var results = new List<DBSnapshot>();  
        var snapshotsPaginator = _amazonRDS.Paginator.DescribeDBSnapshots(  
            new DescribeDBSnapshotsRequest()  
            {  
                DBInstanceIdentifier = dbInstanceIdentifier  
            });  
  
        // Get the entire list using the paginator.  
        await foreach (var snapshots in snapshotsPaginator.DBSnapshots)  
        {  
            results.Add(snapshots);  
        }  
        return results;  
    }  
}
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [CreateDBInstance](#)
  - [CreateDBParameterGroup](#)
  - [CreateDBSnapshot](#)
  - [DeleteDBInstance](#)
  - [DeleteDBParameterGroup](#)
  - [DescribeDBEngineVersions](#)
  - [DescribeDBInstances](#)
  - [DescribeDBParameterGroups](#)
  - [DescribeDBParameters](#)
  - [DescribeDBSnapshots](#)
  - [DescribeOrderableDBInstanceOptions](#)
  - [ModifyDBParameterGroup](#)

Java

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run multiple operations.

```
/**  
 * Before running this Java (v2) code example, set up your development environment,  
 * including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * This Java example performs the following tasks:  
 *  
 * 1. Returns a list of the available DB engines.  
 * 2. Selects an engine family and create a custom DB parameter group.  
 * 3. Gets the parameter groups.  
 * 4. Gets parameters in the group.  
 * 5. Modifies the auto_increment_offset parameter.  
 * 6. Gets and displays the updated parameters.  
 * 7. Gets a list of allowed engine versions.  
 * 8. Gets a list of micro instance classes available for the selected engine.  
 * 9. Creates an RDS database instance that contains a MySql database and uses the  
 *    parameter group.  
 * 10. Waits for the DB instance to be ready and prints out the connection endpoint  
 *     value.  
 * 11. Creates a snapshot of the DB instance.  
 * 12. Waits for an RDS DB snapshot to be ready.  
 * 13. Deletes the RDS DB instance.  
 * 14. Deletes the parameter group.  
 */  
public class RDSScenario {  
  
    public static long sleepTime = 20;  
    public static final String DASHES = new String(new char[80]).replace("\0",  
        "-");
```

```
public static void main(String[] args) throws InterruptedException {  
  
    final String usage = "\n" +  
        "Usage:\n" +  
        "  <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>  
<dbName> <masterUsername> <masterUserPassword> <dbSnapshotIdentifier>\n\n" +  
        "Where:\n" +  
        "  dbGroupName - The database group name. \n"+  
        "  dbParameterGroupFamily - The database parameter group name (for  
example, mysql8.0).\n"+  
        "  dbInstanceIdentifier - The database instance identifier \n"+  
        "  dbName - The database name. \n"+  
        "  masterUsername - The master user name. \n"+  
        "  masterUserPassword - The password that corresponds to the master  
user name. \n"+  
        "  dbSnapshotIdentifier - The snapshot identifier. \n" ;  
  
    if (args.length != 7) {  
        System.out.println(usage);  
        System.exit(1);  
    }  
  
    String dbGroupName = args[0];  
    String dbParameterGroupFamily = args[1];  
    String dbInstanceIdentifier = args[2];  
    String dbName = args[3];  
    String masterUsername = args[4];  
    String masterUserPassword = args[5];  
    String dbSnapshotIdentifier = args[6];  
  
    Region region = Region.US_WEST_2;  
    RdsClient rdsClient = RdsClient.builder()  
        .region(region)  
        .credentialsProvider(ProfileCredentialsProvider.create())  
        .build();  
    System.out.println(DASHES);  
    System.out.println("Welcome to the Amazon RDS example scenario.");  
    System.out.println(DASHES);  
  
    System.out.println(DASHES);  
    System.out.println("1. Return a list of the available DB engines");  
    describeDBEngines(rdsClient);  
    System.out.println(DASHES);  
  
    System.out.println(DASHES);  
    System.out.println("2. Create a custom parameter group");  
    createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);  
    System.out.println(DASHES);  
  
    System.out.println(DASHES);  
    System.out.println("3. Get the parameter group");  
    describeDbParameterGroups(rdsClient, dbGroupName);  
    System.out.println(DASHES);  
  
    System.out.println(DASHES);  
    System.out.println("4. Get the parameters in the group");  
    describeDbParameters(rdsClient, dbGroupName, 0);  
    System.out.println(DASHES);  
  
    System.out.println(DASHES);  
    System.out.println("5. Modify the auto_increment_offset parameter");  
    modifyDBParas(rdsClient, dbGroupName);  
    System.out.println(DASHES);  
  
    System.out.println(DASHES);  
    System.out.println("6. Display the updated value");  
}
```

```
        describeDbParameters(rdsClient, dbGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get a list of micro instance classes available for
the selected engine");
        getMicroInstances(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Create an RDS database instance that contains a
MySQL database and uses the parameter group");
        String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
dbInstanceIdentifier, dbName, masterUsername, masterUserPassword);
        System.out.println("The ARN of the new database is "+dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Wait for DB instance to be ready" );
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a snapshot of the DB instance");
        createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB snapshot to be ready" );
        waitForSnapshotReady(rdsClient, dbInstanceIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the DB instance" );
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the parameter group");
        deleteParaGroup(rdsClient, dbGroupName, dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed." );
        System.out.println(DASHES);

        rdsClient.close();
    }

    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while
database exists.
    public static void deleteParaGroup( RdsClient rdsClient, String dbGroupName,
String dbARN) throws InterruptedException {
        try {
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN ;

            // Make sure that the database has been deleted.
```

```

        while (!isDataDel) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            isDataDel = false ;
            didFind = false;
            int index = 1;
            for (DBInstance instance: instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(dbARN) == 0) {
                    System.out.println(dbARN + " still exists");
                    didFind = true ;
                }
                if ((index == listSize) && (!didFind)) {
                    // Went through the entire list and did not find the
database ARN.
                    isDataDel = true;
                }
                Thread.sleep(sleepTime * 1000);
                index++;
            }
        }

        // Delete the para group.
        DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
    .dbParameterGroupName(dbGroupName)
    .build();

        rdsClient.deleteDBParameterGroup(parameterGroupRequest);
        System.out.println(dbGroupName +" was deleted.");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Delete the DB instance.
public static void deleteDatabaseInstance( RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
    .dbInstanceIdentifier(dbInstanceIdentifier)
    .deleteAutomatedBackups(true)
    .skipFinalSnapshot(true)
    .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.print("The status of the database is " +
response.dbInstance().dbInstanceState());
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the snapshot instance is available.
public static void waitForSnapshotReady(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
        boolean snapshotReady = false;

```

```
String snapshotReadyStr;
System.out.println("Waiting for the snapshot to become available.");

DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
    .dbSnapshotIdentifier(dbSnapshotIdentifier)
    .dbInstanceIdentifier(dbInstanceIdentifier)
    .build();

while (!snapshotReady) {
    DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
    List<DBSnapshot> snapshotList = response.dbSnapshots();
    for (DBSnapshot snapshot : snapshotList) {
        snapshotReadyStr = snapshot.status();
        if (snapshotReadyStr.contains("available")) {
            snapshotReady = true;
        } else {
            System.out.print(".");
            Thread.sleep(sleepTime * 1000);
        }
    }
}

System.out.println("The Snapshot is available!");
} catch (RdsException | InterruptedException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
try {
    CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
    .dbInstanceIdentifier(dbInstanceIdentifier)
    .dbSnapshotIdentifier(dbSnapshotIdentifier)
    .build();

    CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
    System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");

    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .build();

        String endpoint="";

```

```

        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceState();
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint().address();
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Create a database instance and return the ARN of the database.
public static String createDatabaseInstance(RdsClient rdsClient,
                                             String dbGroupName,
                                             String dbInstanceIdentifier,
                                             String dbName,
                                             String masterUsername,
                                             String masterUserPassword) {

    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .allocatedStorage(100)
        .dbName(dbName)
        .dbParameterGroupName(dbGroupName)
        .engine("mysql")
        .dbInstanceClass("db.m4.large")
        .engineVersion("8.0")
        .storageType("standard")
        .masterUsername(masterUsername)
        .masterUserPassword(masterUserPassword)
        .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceState());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }

    return "";
}

// Get a list of micro instances.
public static void getMicroInstances(RdsClient rdsClient) {
    try {
        DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest.builder()

```

```
.engine("mysql")
.build();

DescribeOrderableDbInstanceOptionsResponse response =
rdsClient.describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
List<OrderableDBInstanceState> orderableDBInstances =
response.orderableDBInstanceState();
for (OrderableDBInstanceState dbInstanceState: orderableDBInstances)
{
    System.out.println("The engine version is "
+dbInstanceState.engineVersion());
    System.out.println("The engine description is "
+dbInstanceState.engine());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
try {
    DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .dbParameterGroupFamily(dbParameterGroupFamily)
    .engine("mysql")
    .build();

    DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
    List<DBEngineVersion> dbEngines = response.dbEngineVersions();
    for (DBEngineVersion dbEngine: dbEngines) {
        System.out.println("The engine version is "
+dbEngine.engineVersion());
        System.out.println("The engine description is "
+dbEngine.dbEngineDescription());
    }

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
try {
    Parameter parameter1 = Parameter.builder()
        .parameterName("auto_increment_offset")
        .applyMethod("immediate")
        .parameterValue("5")
        .build();

    List<Parameter> paraList = new ArrayList<>();
    paraList.add(parameter1);
    ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
    .dbParameterGroupName(dbGroupName)
    .parameters(paraList)
    .build();

    ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
```

```
        System.out.println("The parameter group "+  
response.dbParameterGroupName() +" was successfully modified");  
  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}  
  
// Retrieve parameters in the group.  
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,  
int flag) {  
    try {  
        DescribeDbParametersRequest dbParameterGroupsRequest;  
        if (flag == 0) {  
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()  
                .dbParameterGroupName(dbGroupName)  
                .build();  
        } else {  
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()  
                .dbParameterGroupName(dbGroupName)  
                .source("user")  
                .build();  
        }  
  
        DescribeDbParametersResponse response =  
rdsClient.describeDBParameters(dbParameterGroupsRequest);  
List<Parameter> dbParameters = response.parameters();  
String paraName;  
for (Parameter para: dbParameters) {  
    // Only print out information about either auto_increment_offset or  
auto_increment_incremet.  
    paraName = para.parameterName();  
    if ( (paraName.compareTo("auto_increment_offset") ==0) ||  
(paraName.compareTo("auto_increment_incremet ") ==0)) {  
        System.out.println("**** The parameter name is " + paraName);  
        System.out.println("**** The parameter value is " +  
para.parameterValue());  
        System.out.println("**** The parameter data type is " +  
para.dataType());  
        System.out.println("**** The parameter description is " +  
para.description());  
        System.out.println("**** The parameter allowed values is " +  
para.allowedValues());  
    }  
}  
  
} catch (RdsException e) {  
    System.out.println(e.getLocalizedMessage());  
    System.exit(1);  
}  
}  
  
public static void describeDbParameterGroups(RdsClient rdsClient, String  
dbGroupName) {  
    try {  
        DescribeDbParameterGroupsRequest groupsRequest =  
DescribeDbParameterGroupsRequest.builder()  
            .dbParameterGroupName(dbGroupName)  
            .maxRecords(20)  
            .build();  
  
        DescribeDbParameterGroupsResponse response =  
rdsClient.describeDBParameterGroups(groupsRequest);  
List<DBParameterGroup> groups = response.dbParameterGroups();  
for (DBParameterGroup group: groups) {
```

```
        System.out.println("The group name is  
"+group.dbParameterGroupName());  
        System.out.println("The group description is  
"+group.description());  
    }  
  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}  
  
public static void createDBParameterGroup(RdsClient rdsClient, String dbGroupName, String dbParameterGroupFamily) {  
    try {  
        CreateDbParameterGroupRequest groupRequest =  
CreateDbParameterGroupRequest.builder()  
            .dbParameterGroupName(dbGroupName)  
            .dbParameterGroupFamily(dbParameterGroupFamily)  
            .description("Created by using the AWS SDK for Java")  
            .build();  
  
        CreateDbParameterGroupResponse response =  
rdsClient.createDBParameterGroup(groupRequest);  
        System.out.println("The group name is "+  
response.dbParameterGroup().dbParameterGroupName());  
  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}  
  
public static void describeDBEngines( RdsClient rdsClient) {  
    try {  
        DescribeDbEngineVersionsRequest engineVersionsRequest =  
DescribeDbEngineVersionsRequest.builder()  
            .defaultOnly(true)  
            .engine("mysql")  
            .maxRecords(20)  
            .build();  
  
        DescribeDbEngineVersionsResponse response =  
rdsClient.describeDBEngineVersions(engineVersionsRequest);  
        List<DBEngineVersion> engines = response.dbEngineVersions();  
  
        // Get all DBEngineVersion objects.  
        for (DBEngineVersion engine0b: engines) {  
            System.out.println("The name of the DB parameter group family for  
the database engine is "+engine0b.dbParameterGroupFamily());  
            System.out.println("The name of the database engine  
"+engine0b.engine());  
            System.out.println("The version number of the database engine  
"+engine0b.engineVersion());  
        }  
  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}  
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [CreateDBInstance](#)
  - [CreateDBParameterGroup](#)
  - [CreateDBSnapshot](#)
  - [DeleteDBInstance](#)
  - [DeleteDBParameterGroup](#)
  - [DescribeDBEngineVersions](#)
  - [DescribeDBInstances](#)
  - [DescribeDBParameterGroups](#)
  - [DescribeDBParameters](#)
  - [DescribeDBSnapshots](#)
  - [DescribeOrderableDBInstanceOptions](#)
  - [ModifyDBParameterGroup](#)

Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
 * Before running this Java V2 code example, set up your development environment,  
 * including your credentials.  
  
 * For more information, see the following documentation topic:  
  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
  
 * This example performs the following tasks:  
  
 * 1. Returns a list of the available DB engines by invoking the  
 *    DescribeDbEngineVersions method.  
 * 2. Selects an engine family and create a custom DB parameter group by invoking the  
 *    createDBParameterGroup method.  
 * 3. Gets the parameter groups by invoking the DescribeDbParameterGroups method.  
 * 4. Gets parameters in the group by invoking the DescribeDbParameters method.  
 * 5. Modifies both the auto_increment_offset and auto_increment_increment parameters  
 *    by invoking the modifyDbParameterGroup method.  
 * 6. Gets and displays the updated parameters.  
 * 7. Gets a list of allowed engine versions by invoking the describeDbEngineVersions  
 *    method.  
 * 8. Gets a list of micro instance classes available for the selected engine.  
 * 9. Creates an RDS database instance that contains a MySQL database and uses the  
 *    parameter group  
 * 10. Waits for DB instance to be ready and print out the connection endpoint value.  
 * 11. Creates a snapshot of the DB instance.  
 * 12. Waits for DB snapshot to be ready.  
 * 13. Deletes the DB instance. rds.DeleteDbInstance.  
 * 14. Deletes the parameter group.  
 */  
  
var sleepTime: Long = 20
```

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
        <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier> <dbName>
        <masterUsername> <masterUserPassword> <dbSnapshotIdentifier>

        Where:
        dbGroupName - The database group name.
        dbParameterGroupFamily - The database parameter group name.
        dbInstanceIdentifier - The database instance identifier.
        dbName - The database name.
        username - The user name.
        userPassword - The password that corresponds to the user name.
        dbSnapshotIdentifier - The snapshot identifier.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }

    val dbGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceIdentifier = args[2]
    val dbName = args[3]
    val username = args[4]
    val userPassword = args[5]
    val dbSnapshotIdentifier = args[6]

    println("1. Return a list of the available DB engines")
    describeDBEngines()

    println("2. Create a custom parameter group")
    createDBParameterGroup(dbGroupName, dbParameterGroupFamily)

    println("3. Get the parameter groups")
    describeDbParameterGroups(dbGroupName)

    println("4. Get the parameters in the group")
    describeDbParameters(dbGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBParas(dbGroupName)

    println("6. Display the updated value")
    describeDbParameters(dbGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedEngines(dbParameterGroupFamily)

    println("8. Get a list of micro instance classes available for the selected
    engine")
    getMicroInstances()

    println("9. Create an RDS database instance that contains a MySql database and
    uses the parameter group")
    val dbARN = createDatabaseInstance(dbGroupName, dbInstanceIdentifier, dbName,
    username, userPassword)
    println("The ARN of the new database is $dbARN")

    println("10. Wait for DB instance to be ready")
    waitForDbInstanceReady(dbInstanceIdentifier)

    println("11. Create a snapshot of the DB instance")
    createDbSnapshot(dbInstanceIdentifier, dbSnapshotIdentifier)
```

```
    println("12. Wait for DB snapshot to be ready")
    waitForSnapshotReady(dbInstanceIdentifier, dbSnapshotIdentifier)

    println("13. Delete the DB instance")
    deleteDbInstance(dbInstanceIdentifier)

    println("14. Delete the parameter group")
    if (dbARN != null) {
        deleteParaGroup(dbGroupName, dbARN)
    }

    println("The Scenario has successfully completed.")
}

suspend fun deleteParaGroup(dbGroupName: String, dbARN: String) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false // Reset this value.
            didFind = false // Reset this value.
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(dbARN) == 0) {
                        println("$dbARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
                        database name.
                        isDataDel = true
                    }
                    index++
                }
            }
        }
        // Delete the para group.
        val parameterGroupRequest = DeleteDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
        }
        rdsClient.deleteDbParameterGroup(parameterGroupRequest)
        println("$dbGroupName was deleted.")
    }
}

suspend fun deleteDbInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest = DeleteDbInstanceRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
        deleteAutomatedBackups = true
        skipFinalSnapshot = true
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
${response.dbInstance?.dbInstanceState}")
    }
}
```

```
}

// Waits until the snapshot instance is available.
suspend fun waitForSnapshotReady(dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest = DescribeDbSnapshotsRequest {
        dbSnapshotIdentifier = dbSnapshotIdentifierVal
        dbInstanceIdentifier = dbInstanceIdentifierVal
    }

    while (!snapshotReady) {
        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbSnapshots(snapshotsRequest)
            val snapshotList: List<DbSnapshot>? = response.dbSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}

// Create an Amazon RDS snapshot.
suspend fun createDbSnapshot(dbInstanceIdentifierVal: String?,
    dbSnapshotIdentifierVal: String?) {
    val snapshotRequest = CreateDbSnapshotRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
        dbSnapshotIdentifier = dbSnapshotIdentifierVal
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbSnapshot(snapshotRequest)
        print("The Snapshot id is ${response.dbSnapshot?.dbiResourceId}")
    }
}

// Waits until the database instance is available.
suspend fun waitForDbInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest = DescribeDbInstancesRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
    }
    var endpoint = ""
    while (!instanceReady) {
        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceState.toString()
                    if (instanceReadyStr.contains("available")) {

```

```

        endpoint = instance.endpoint?.address.toString()
        instanceReady = true
    } else {
        print(".")
        delay(sleepTime * 1000)
    }
}
}
}
}
println("Database instance is available! The connection endpoint is $endpoint")
}

// Create a database instance and return the ARN of the database.
suspend fun createDatabaseInstance(dbGroupNameVal: String?,
    dbInstanceIdentifierVal: String?, dbNameVal: String?, masterUsernameVal: String?,
    masterUserPasswordVal: String?): String? {
    val instanceRequest = CreateDbInstanceRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
        allocatedStorage = 100
        dbName = dbNameVal
        dbParameterGroupName = dbGroupNameVal
        engine = "mysql"
        dbInstanceClass = "db.m4.large"
        engineVersion = "8.0"
        storageType = "standard"
        masterUsername = masterUsernameVal
        masterUserPassword = masterUserPasswordVal
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceState}")
        return response.dbInstance?.dbInstanceArn
    }
}

// Get a list of micro instances.
suspend fun getMicroInstances() {
    val dbInstanceOptionsRequest = DescribeOrderableDbInstanceOptionsRequest {
        engine = "mysql"
    }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
            rdsClient.describeOrderableDbInstanceOptions(dbInstanceOptionsRequest)
        val orderableDBInstances = response.orderableDbInstanceOptions
        if (orderableDBInstances != null) {
            for (dbInstanceOption in orderableDBInstances) {
                println("The engine version is ${dbInstanceOption.engineVersion}")
                println("The engine description is ${dbInstanceOption.engine}")
            }
        }
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest = DescribeDbEngineVersionsRequest {
        dbParameterGroupFamily = dbParameterGroupFamilyVal
        engine = "mysql"
    }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        val dbEngines: List<DbEngineVersion>? = response.dbEngineVersions
        if (dbEngines != null) {
            for (dbEngine in dbEngines) {

```

```
        println("The engine version is ${dbEngine.engineVersion}")
        println("The engine description is
${dbEngine.dbEngineDescription}")
    }
}
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBParas(dbGroupName: String) {
    val parameter1 = Parameter {
        parameterName = "auto_increment_offset"
        applyMethod = ApplyMethod.Immediate
        parameterValue = "5"
    }

    val paraList: ArrayList<Parameter> = ArrayList()
    paraList.add(parameter1)
    val groupRequest = ModifyDbParameterGroupRequest {
        dbParameterGroupName = dbGroupName
        parameters = paraList
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbParameterGroup(groupRequest)
        println("The parameter group ${response.dbParameterGroupName} was
successfully modified")
    }
}

// Retrieve parameters in the group.
suspend fun describeDbParameters(dbGroupName: String?, flag: Int) {
    val dbParameterGroupsRequest: DescribeDbParametersRequest
    dbParameterGroupsRequest = if (flag == 0) {
        DescribeDbParametersRequest {
            dbParameterGroupName = dbGroupName
        }
    } else {
        DescribeDbParametersRequest {
            dbParameterGroupName = dbGroupName
            source = "user"
        }
    }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameters(dbParameterGroupsRequest)
        val dbParameters: List<Parameter>? = response.parameters
        var paraName: String
        if (dbParameters != null) {
            for (para in dbParameters) {
                // Only print out information about either auto_increment_offset or
auto_increment_increment.
                paraName = para.parameterName.toString()
                if (paraName.compareTo("auto_increment_offset") == 0 ||
paraName.compareTo("auto_increment_increment ") == 0) {
                    println("**** The parameter name is $paraName")
                    System.out.println("**** The parameter value is
${para.parameterValue}")
                    System.out.println("**** The parameter data type is
${para.dataType}")
                    System.out.println("**** The parameter description is
${para.description}")
                    System.out.println("**** The parameter allowed values is
${para.allowedValues}")
                }
            }
        }
    }
}
```

```

        }

    }

    suspend fun describeDbParameterGroups(dbGroupName: String?) {
        val groupsRequest = DescribeDbParameterGroupsRequest {
            dbParameterGroupName = dbGroupName
            maxRecords = 20
        }
        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbParameterGroups(groupsRequest)
            val groups = response.dbParameterGroups
            if (groups != null) {
                for (group in groups) {
                    println("The group name is ${group.dbParameterGroupName}")
                    println("The group description is ${group.description}")
                }
            }
        }
    }

    // Create a parameter group.
    suspend fun createDBParameterGroup(dbGroupName: String?, dbParameterGroupFamilyVal: String?) {
        val groupRequest = CreateDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.createDbParameterGroup(groupRequest)
            println("The group name is
${response.dbParameterGroup?.dbParameterGroupName}")
        }
    }

    // Returns a list of the available DB engines.
    suspend fun describeDBEngines() {
        val engineVersionsRequest = DescribeDbEngineVersionsRequest {
            defaultOnly = true
            engine = "mysql"
            maxRecords = 20
        }

        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
            val engines: List<DbEngineVersion>? = response.dbEngineVersions

            // Get all DbEngineVersion objects.
            if (engines != null) {
                for (engineOb in engines) {
                    println("The name of the DB parameter group family for the database
engine is ${engineOb.dbParameterGroupFamily}.")
                    println("The name of the database engine ${engineOb.engine}.")
                    println("The version number of the database engine
${engineOb.engineVersion}")
                }
            }
        }
    }
}

```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - [CreateDBInstance](#)

- [CreateDBParameterGroup](#)
- [CreateDBSnapshot](#)
- [DeleteDBInstance](#)
- [DeleteDBParameterGroup](#)
- [DescribeDBEngineVersions](#)
- [DescribeDBInstances](#)
- [DescribeDBParameterGroups](#)
- [DescribeDBParameters](#)
- [DescribeDBSnapshots](#)
- [DescribeOrderableDBInstanceOptions](#)
- [ModifyDBParameterGroup](#)

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
class RdsInstanceScenario:  
    """Runs a scenario that shows how to get started using Amazon RDS DB  
    instances."""  
    def __init__(self, instance_wrapper):  
        """  
        :param instance_wrapper: An object that wraps Amazon RDS DB instance  
        actions.  
        """  
        self.instance_wrapper = instance_wrapper  
  
    def create_parameter_group(self, parameter_group_name, db_engine):  
        """  
        Shows how to get available engine versions for a specified database engine  
        and  
        create a DB parameter group that is compatible with a selected engine  
        family.  
  
        :param parameter_group_name: The name given to the newly created parameter  
        group.  
        :param db_engine: The database engine to use as a basis.  
        :return: The newly created parameter group.  
        """  
        print(f"Checking for an existing DB instance parameter group named  
        {parameter_group_name}.")  
        parameter_group =  
        self.instance_wrapper.get_parameter_group(parameter_group_name)  
        if parameter_group is None:  
            print(f"Getting available database engine versions for {db_engine}.")  
            engine_versions = self.instance_wrapper.get_engine_versions(db_engine)  
            families = list({ver['DBParameterGroupFamily'] for ver in  
            engine_versions})  
            family_index = q.choose("Which family do you want to use? ", families)  
            print(f"Creating a parameter group.")  
            self.instance_wrapper.create_parameter_group(  
                parameter_group_name, families[family_index], 'Example parameter  
group.')  
        """
```

```

        parameter_group =
    self.instance_wrapper.get_parameter_group(parameter_group_name)
    print(f"Parameter group {parameter_group['DBParameterGroupName']}:")
    pp(parameter_group)
    print('*'*88)
    return parameter_group

def update_parameters(self, parameter_group_name):
    """
    Shows how to get the parameters contained in a custom parameter group and
    update some of the parameter values in the group.

    :param parameter_group_name: The name of the parameter group to query and
    modify.
    """
    print("Let's set some parameter values in your parameter group.")
    auto_inc_parameters = self.instance_wrapper.get_parameters(
        parameter_group_name, name_prefix='auto_increment')
    update_params = []
    for auto_inc in auto_inc_parameters:
        if auto_inc['IsModifiable'] and auto_inc['DataType'] == 'integer':
            print(f"The {auto_inc['ParameterName']} parameter is described
as:")
            print(f"\t{auto_inc['Description']}")
            param_range = auto_inc['AllowedValues'].split('-')
            auto_inc['ParameterValue'] = str(q.ask(
                f"Enter a value between {param_range[0]} and {param_range[1]}:
",
                q.is_int, q.in_range(int(param_range[0]),
int(param_range[1]))))
            update_params.append(auto_inc)
    self.instance_wrapper.update_parameters(parameter_group_name,
update_params)
    print("You can get a list of parameters you've set by specifying a source
of 'user'.")
    user_parameters =
self.instance_wrapper.get_parameters(parameter_group_name, source='user')
    pp(user_parameters)
    print('*'*88)

def create_instance(self, instance_name, db_name, db_engine, parameter_group):
    """
    Shows how to create a DB instance that contains a database of a specified
    type and is configured to use a custom DB parameter group.

    :param instance_name: The name given to the newly created DB instance.
    :param db_name: The name given to the created database.
    :param db_engine: The engine of the created database.
    :param parameter_group: The parameter group that is associated with the DB
    instance.
    :return: The newly created DB instance.
    """
    print("Checking for an existing DB instance.")
    db_inst = self.instance_wrapper.get_db_instance(instance_name)
    if db_inst is None:
        print("Let's create a DB instance.")
        admin_username = q.ask("Enter an administrator user name for the
database: ", q.non_empty)
        admin_password = q.ask(
            "Enter a password for the administrator (at least 8 characters): ",
q.non_empty)
        engine_versions = self.instance_wrapper.get_engine_versions(
            db_engine, parameter_group['DBParameterGroupFamily'])
        engine_choices = [ver['EngineVersion'] for ver in engine_versions]
        print("The available engines for your parameter group are:")

```

```

        engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
        engine_selection = engine_versions[engine_index]
        print("The available micro DB instance classes for your database engine
are:")
        inst_opts = self.instance_wrapper.get_orderable_instances(
            engine_selection['Engine'], engine_selection['EngineVersion'])
        inst_choices = list([opt['DBInstanceClass'] for opt in inst_opts if
'micro' in opt['DBInstanceClass']])
        inst_index = q.choose("Which micro DB instance class do you want to
use? ", inst_choices)
        group_name = parameter_group['DBParameterGroupName']
        storage_type = 'standard'
        allocated_storage = 5
        print(f"Creating a DB instance named {instance_name} and database
{db_name}.\n"
              f"The DB instance is configured to use your custom parameter
group {group_name},\n"
              f"selected engine {engine_selection['EngineVersion']},\n"
              f"selected DB instance class {inst_choices[inst_index]},"
              f"and {allocated_storage} GiB of {storage_type} storage.\n"
              f"This typically takes several minutes.")
        db_inst = self.instance_wrapper.create_db_instance(
            db_name, instance_name, group_name, engine_selection['Engine'],
            engine_selection['EngineVersion'], inst_choices[inst_index],
            storage_type,
            allocated_storage, admin_username, admin_password)
        while db_inst.get('DBInstanceState') != 'available':
            wait(10)
            db_inst = self.instance_wrapper.get_db_instance(instance_name)
        print("Instance data:")
        pp(db_inst)
        print('*'*88)
        return db_inst

    @staticmethod
    def display_connection(db_inst):
        """
        Displays connection information about a DB instance and tips on how to
        connect to it.

        :param db_inst: The DB instance to display.
        """
        print("You can now connect to your database using your favorite MySql
client.\n"
              "One way to connect is by using the 'mysql' shell on an Amazon EC2
instance\n"
              "that is running in the same VPC as your DB instance. Pass the
endpoint,\n"
              "port, and administrator user name to 'mysql' and enter your password
\n"
              "when prompted:\n")
        print(f"\nmysql -h {db_inst['Endpoint']['Address']} -P
{db_inst['Endpoint']['Port']} "
              f"-u {db_inst['MasterUsername']} -p\n")
        print("For more information, see the User Guide for Amazon RDS:\n"
              "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL")
        print('*'*88)

    def create_snapshot(self, instance_name):
        """
        Shows how to create a DB instance snapshot and wait until it's available.

        :param instance_name: The name of a DB instance to snapshot.
        """

```

```

        if q.ask("Do you want to create a snapshot of your DB instance (y/n)? ",
q.is_yesno):
    snapshot_id = f"{instance_name}-{uuid.uuid4()}"
    print(f"Creating a snapshot named {snapshot_id}. This typically takes a
few minutes.")
    snapshot = self.instance_wrapper.create_snapshot(snapshot_id,
instance_name)
    while snapshot.get('Status') != 'available':
        wait(10)
        snapshot = self.instance_wrapper.get_snapshot(snapshot_id)
    pp(snapshot)
    print('*'*88)

def cleanup(self, db_inst, parameter_group_name):
"""
Shows how to clean up a DB instance and parameter group.
Before the parameter group can be deleted, all associated DB instances must
first
be deleted.

:param db_inst: The DB instance to delete.
:param parameter_group_name: The DB parameter group to delete.
"""
if q.ask(
        "\nDo you want to delete the DB instance and parameter group (y/n)?",
        q.is_yesno):
    print(f"Deleting DB instance {db_inst['DBInstanceIdentifier']}.")

self.instance_wrapper.delete_db_instance(db_inst['DBInstanceIdentifier'])
    print("Waiting for the DB instance to delete. This typically takes
several minutes.")
    while db_inst is not None:
        wait(10)
        db_inst =
self.instance_wrapper.get_db_instance(db_inst['DBInstanceIdentifier'])
    print(f"Deleting parameter group {parameter_group_name}.")
    self.instance_wrapper.delete_parameter_group(parameter_group_name)

def run_scenario(
        self, db_engine, parameter_group_name, instance_name, db_name):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s:
%(message)s')

    print('*'*88)
    print("Welcome to the Amazon Relational Database Service (Amazon RDS)\n"
          "get started with DB instances demo.")
    print('*'*88)

    parameter_group = self.create_parameter_group(parameter_group_name,
db_engine)
    self.update_parameters(parameter_group_name)
    db_inst = self.create_instance(instance_name, db_name, db_engine,
parameter_group)
    self.display_connection(db_inst)
    self.create_snapshot(instance_name)
    self.cleanup(db_inst, parameter_group_name)

    print("\nThanks for watching!")
    print('*'*88)

if __name__ == '__main__':
    try:
        scenario = RdsInstanceScenario(InstanceWrapper.from_client())
        scenario.run_scenario()

```

```
'mysql', 'doc-example-parameter-group', 'doc-example-instance',
'docexampledb')
except Exception:
    logging.exception("Something went wrong with the demo.")
```

Define functions that are called by the scenario to manage Amazon RDS actions.

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name)
            parameter_group = response['DBParameterGroups'][0]
        except ClientError as err:
            if err.response['Error']['Code'] == 'DBParameterGroupNotFound':
                logger.info("Parameter group %s does not exist.",
                            parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response['Error']['Code'], err.response['Error']
                    ['Message'])
                raise
        else:
            return parameter_group

    def create_parameter_group(self, parameter_group_name, parameter_group_family,
                               description):
        """
        Creates a DB parameter group that is based on the specified parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter group.
        :param parameter_group_family: The family that is used as the basis of the
                                       new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
```

```

        DBParameterGroupFamily=parameter_group_family,
        Description=description)
    except ClientError as err:
        logger.error(
            "Couldn't create parameter group %s. Here's why: %s: %s",
            parameter_group_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response

def delete_parameter_group(self, parameter_group_name):
    """
    Deletes a DB parameter group.

    :param parameter_group_name: The name of the parameter group to delete.
    :return: Data about the parameter group.
    """
    try:
        self.rds_client.delete_db_parameter_group(
            DBParameterGroupName=parameter_group_name)
    except ClientError as err:
        logger.error(
            "Couldn't delete parameter group %s. Here's why: %s: %s",
            parameter_group_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def get_parameters(self, parameter_group_name, name_prefix='', source=None):
    """
    Gets the parameters that are contained in a DB parameter group.

    :param parameter_group_name: The name of the parameter group to query.
    :param name_prefix: When specified, the retrieved list of parameters is
    filtered
        to contain only parameters that start with this prefix.
    :param source: When specified, only parameters from this source are
    retrieved.
        For example, a source of 'user' retrieves only parameters
    that
        were set by a user.
    :return: The list of requested parameters.
    """
    try:
        kwargs = {'DBParameterGroupName': parameter_group_name}
        if source is not None:
            kwargs['Source'] = source
        parameters = []
        paginator = self.rds_client.get_paginator('describe_db_parameters')
        for page in paginator.paginate(**kwargs):
            parameters += [
                p for p in page['Parameters'] if
                p['ParameterName'].startswith(name_prefix)]
    except ClientError as err:
        logger.error(
            "Couldn't get parameters for %s. Here's why: %s: %s",
            parameter_group_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return parameters

def update_parameters(self, parameter_group_name, update_parameters):
    """
    Updates parameters in a custom DB parameter group.

```

```
:param parameter_group_name: The name of the parameter group to update.  
:param update_parameters: The parameters to update in the group.  
:return: Data about the modified parameter group.  
"""  
    try:  
        response = self.rds_client.modify_db_parameter_group(  
            DBParameterGroupName=parameter_group_name,  
            Parameters=update_parameters)  
    except ClientError as err:  
        logger.error(  
            "Couldn't update parameters in %s. Here's why: %s: %s",  
            parameter_group_name,  
            err.response['Error']['Code'], err.response['Error']['Message'])  
        raise  
    else:  
        return response  
  
def create_snapshot(self, snapshot_id, instance_id):  
    """  
    Creates a snapshot of a DB instance.  
  
    :param snapshot_id: The ID to give the created snapshot.  
    :param instance_id: The ID of the DB instance to snapshot.  
    :return: Data about the newly created snapshot.  
    """  
    try:  
        response = self.rds_client.create_db_snapshot(  
            DBSnapshotIdentifier=snapshot_id, DBInstanceIdentifier=instance_id)  
        snapshot = response['DBSnapshot']  
    except ClientError as err:  
        logger.error(  
            "Couldn't create snapshot of %s. Here's why: %s: %s", instance_id,  
            err.response['Error']['Code'], err.response['Error']['Message'])  
        raise  
    else:  
        return snapshot  
  
def get_snapshot(self, snapshot_id):  
    """  
    Gets a DB instance snapshot.  
  
    :param snapshot_id: The ID of the snapshot to retrieve.  
    :return: The retrieved snapshot.  
    """  
    try:  
        response = self.rds_client.describe_db_snapshots(  
            DBSnapshotIdentifier=snapshot_id)  
        snapshot = response['DBSnapshots'][0]  
    except ClientError as err:  
        logger.error(  
            "Couldn't get snapshot %s. Here's why: %s: %s", snapshot_id,  
            err.response['Error']['Code'], err.response['Error']['Message'])  
        raise  
    else:  
        return snapshot  
  
def get_engine_versions(self, engine, parameter_group_family=None):  
    """  
    Gets database engine versions that are available for the specified engine  
    and parameter group family.  
  
    :param engine: The database engine to look up.  
    :param parameter_group_family: When specified, restricts the returned list  
        of engine versions to those that are compatible  
        with  
    """
```

```
        this parameter group family.  
:return: The list of database engine versions.  
"""  
    try:  
        kwargs = {'Engine': engine}  
        if parameter_group_family is not None:  
            kwargs['DBParameterGroupFamily'] = parameter_group_family  
        response = self.rds_client.describe_db_engine_versions(**kwargs)  
        versions = response['DBEngineVersions']  
    except ClientError as err:  
        logger.error(  
            "Couldn't get engine versions for %s. Here's why: %s: %s", engine,  
            err.response['Error']['Code'], err.response['Error']['Message'])  
        raise  
    else:  
        return versions  
  
def get_orderable_instances(self, db_engine, db_engine_version):  
    """  
    Gets DB instance options that can be used to create DB instances that are  
    compatible with a set of specifications.  
  
    :param db_engine: The database engine that must be supported by the DB  
    instance.  
    :param db_engine_version: The engine version that must be supported by the  
    DB instance.  
    :return: The list of DB instance options that can be used to create a  
    compatible DB instance.  
    """  
    try:  
        inst_opts = []  
        paginator =  
self.rds_client.getPaginator('describe_orderable_db_instance_options')  
        for page in paginator.paginate(Engine=db_engine,  
        EngineVersion=db_engine_version):  
            inst_opts += page['OrderableDBInstanceOptions']  
    except ClientError as err:  
        logger.error(  
            "Couldn't get orderable DB instances. Here's why: %s: %s",  
            err.response['Error']['Code'], err.response['Error']['Message'])  
        raise  
    else:  
        return inst_opts  
  
def get_db_instance(self, instance_id):  
    """  
    Gets data about a DB instance.  
  
    :param instance_id: The ID of the DB instance to retrieve.  
    :return: The retrieved DB instance.  
    """  
    try:  
        response = self.rds_client.describe_db_instances(  
            DBInstanceIdentifier=instance_id)  
        db_inst = response['DBInstances'][0]  
    except ClientError as err:  
        if err.response['Error']['Code'] == 'DBInstanceNotFound':  
            logger.info("Instance %s does not exist.", instance_id)  
        else:  
            logger.error(  
                "Couldn't get DB instance %s. Here's why: %s: %s", instance_id,  
                err.response['Error']['Code'], err.response['Error']  
                ['Message'])  
            raise  
    else:  
        return db_inst
```

```
def create_db_instance(
    self, db_name, instance_id, parameter_group_name, db_engine,
    db_engine_version,
        instance_class, storage_type, allocated_storage, admin_name,
    admin_password):
    """
    Creates a DB instance.

    :param db_name: The name of the database that is created in the DB
    instance.
    :param instance_id: The ID to give the newly created DB instance.
    :param parameter_group_name: A parameter group to associate with the DB
    instance.
    :param db_engine: The database engine of a database to create in the DB
    instance.
    :param db_engine_version: The engine version for the created database.
    :param instance_class: The DB instance class for the newly created DB
    instance.
    :param storage_type: The storage type of the DB instance.
    :param allocated_storage: The amount of storage allocated on the DB
    instance, in GiBs.
    :param admin_name: The name of the admin user for the created database.
    :param admin_password: The admin password for the created database.
    :return: Data about the newly created DB instance.
    """
    try:
        response = self.rds_client.create_db_instance(
            DBName=db_name,
            DBInstanceIdentifier=instance_id,
            DBParameterGroupName=parameter_group_name,
            Engine=db_engine,
            EngineVersion=db_engine_version,
            DBInstanceClass=instance_class,
            StorageType=storage_type,
            AllocatedStorage=allocated_storage,
            MasterUsername=admin_name,
            MasterUserPassword=admin_password)
        db_inst = response['DBInstance']
    except ClientError as err:
        logger.error(
            "Couldn't create DB instance %s. Here's why: %s: %s",
            instance_id,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return db_inst

def delete_db_instance(self, instance_id):
    """
    Deletes a DB instance.

    :param instance_id: The ID of the DB instance to delete.
    :return: Data about the deleted DB instance.
    """
    try:
        response = self.rds_client.delete_db_instance(
            DBInstanceIdentifier=instance_id, SkipFinalSnapshot=True,
            DeleteAutomatedBackups=True)
        db_inst = response['DBInstance']
    except ClientError as err:
        logger.error(
            "Couldn't delete DB instance %s. Here's why: %s: %s",
            instance_id,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return db_inst
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateDBInstance](#)
  - [CreateDBParameterGroup](#)
  - [CreateDBSnapshot](#)
  - [DeleteDBInstance](#)
  - [DeleteDBParameterGroup](#)
  - [DescribeDBEngineVersions](#)
  - [DescribeDBInstances](#)
  - [DescribeDBParameterGroups](#)
  - [DescribeDBParameters](#)
  - [DescribeDBSnapshots](#)
  - [DescribeOrderableDBInstanceOptions](#)
  - [ModifyDBParameterGroup](#)

## Cross-service examples for Amazon RDS using AWS SDKs

The following code examples show how to use Amazon Relational Database Service with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Create a lending library REST API \(p. 1501\)](#)
- [Create an Aurora Serverless work item tracker \(p. 1502\)](#)

### Create a lending library REST API

The following code example shows how to create a lending library where patrons can borrow and return books by using a REST API backed by an Amazon Aurora database.

#### Python

##### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with the Amazon Relational Database Service (Amazon RDS) API and AWS Chalice to create a REST API backed by an Amazon Aurora database. The web service is fully serverless and represents a simple lending library where patrons can borrow and return books. Learn how to:

- Create and manage a serverless Aurora database cluster.
- Use AWS Secrets Manager to manage database credentials.
- Implement a data storage layer that uses Amazon RDS to move data into and out of the database.
- Use AWS Chalice to deploy a serverless REST API to Amazon API Gateway and AWS Lambda.
- Use the Requests package to send requests to the web service.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- API Gateway
- Lambda
- Amazon RDS
- Secrets Manager

## Create an Aurora Serverless work item tracker

The following code examples show how to create a web application that tracks work items in an Amazon Aurora Serverless database and uses Amazon Simple Email Service (Amazon SES) to send reports.

.NET

### AWS SDK for .NET

Shows how to use the AWS SDK for .NET to create a web application that tracks work items in an Amazon Aurora database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with a RESTful .NET backend.

- Integrate a React web application with AWS services.
- List, add, update, and delete items in an Aurora table.
- Send an email report of filtered work items using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

C++

### SDK for C++

Shows how to create a web application that tracks and reports on work items stored in an Amazon Aurora Serverless database.

For complete source code and instructions on how to set up a C++ REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

## Java

### SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

For complete source code and instructions on how to set up and run an example that uses the JDBC API, see the full example on [GitHub](#).

#### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

## JavaScript

### SDK for JavaScript V3

Shows how to use the AWS SDK for JavaScript (v3) to create a web application that tracks work items in an Amazon Aurora database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with an Express Node.js backend.

- Integrate a React.js web application with AWS services.
- List, add, and update items in an Aurora table.
- Send an email report of filtered work items by using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

PHP

#### SDK for PHP

Shows how to use the AWS SDK for PHP to create a web application that tracks work items in an Amazon RDS database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with a RESTful PHP backend.

- Integrate a React.js web application with AWS services.
- List, add, update, and delete items in an Amazon RDS table.
- Send an email report of filtered work items using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Python

#### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) to create a REST service that tracks work items in an Amazon Aurora Serverless database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses the Flask web framework to handle HTTP routing and integrates with a React webpage to present a fully functional web application.

- Build a Flask REST service that integrates with AWS services.
- Read, write, and update work items that are stored in an Aurora Serverless database.
- Create an AWS Secrets Manager secret that contains database credentials and use it to authenticate calls to the database.
- Use Amazon SES to send email reports of work items.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service

- Amazon SES

## Code examples for Amazon RDS Data Service using AWS SDKs

The following code examples show how to use Amazon Relational Database Service Data Service with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon RDS Data Service using AWS SDKs \(p. 1505\)](#)
  - [Run an SQL statement using an AWS SDK \(p. 1505\)](#)
- [Cross-service examples for Amazon RDS Data Service using AWS SDKs \(p. 1506\)](#)
  - [Create an Aurora Serverless work item tracker \(p. 1506\)](#)

## Actions for Amazon RDS Data Service using AWS SDKs

The following code examples show how to use Amazon Relational Database Service Data Service with AWS SDKs. Each example calls an individual service function.

### Examples

- [Run an SQL statement using an AWS SDK \(p. 1505\)](#)

## Run an SQL statement using an AWS SDK

The following code example shows how to run an SQL statement.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn query_cluster(
    client: &Client,
    cluster_arn: &str,
    query: &str,
    secret_arn: &str,
) -> Result<(), Error> {
    let st = client
        .execute_statement()
        .resource_arn(cluster_arn)
        .database("postgres") // Do not confuse this with db instance name
        .sql(query)
        .secret_arn(secret_arn);
```

```
    let result = st.send().await?;

    println!("{:?}", result);
    println!();

    Ok(())
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Rust API reference*.

## Cross-service examples for Amazon RDS Data Service using AWS SDKs

The following code examples show how to use Amazon Relational Database Service Data Service with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Create an Aurora Serverless work item tracker \(p. 1506\)](#)

## Create an Aurora Serverless work item tracker

The following code examples show how to create a web application that tracks work items in an Amazon Aurora Serverless database and uses Amazon Simple Email Service (Amazon SES) to send reports.

### .NET

#### AWS SDK for .NET

Shows how to use the AWS SDK for .NET to create a web application that tracks work items in an Amazon Aurora database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with a RESTful .NET backend.

- Integrate a React web application with AWS services.
- List, add, update, and delete items in an Aurora table.
- Send an email report of filtered work items using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

### C++

#### SDK for C++

Shows how to create a web application that tracks and reports on work items stored in an Amazon Aurora Serverless database.

For complete source code and instructions on how to set up a C++ REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Java

#### SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

For complete source code and instructions on how to set up and run an example that uses the JDBC API, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

JavaScript

#### SDK for JavaScript V3

Shows how to use the AWS SDK for JavaScript (v3) to create a web application that tracks work items in an Amazon Aurora database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with an Express Node.js backend.

- Integrate a React.js web application with AWS services.
- List, add, and update items in an Aurora table.
- Send an email report of filtered work items by using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service

- Amazon SES

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

#### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

PHP

### SDK for PHP

Shows how to use the AWS SDK for PHP to create a web application that tracks work items in an Amazon RDS database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with a RESTful PHP backend.

- Integrate a React.js web application with AWS services.
- List, add, update, and delete items in an Amazon RDS table.
- Send an email report of filtered work items using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Python

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) to create a REST service that tracks work items in an Amazon Aurora Serverless database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses the Flask web framework to handle HTTP routing and integrates with a React webpage to present a fully functional web application.

- Build a Flask REST service that integrates with AWS services.
- Read, write, and update work items that are stored in an Aurora Serverless database.
- Create an AWS Secrets Manager secret that contains database credentials and use it to authenticate calls to the database.
- Use Amazon SES to send email reports of work items.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

## Code examples for Amazon Redshift using AWS SDKs

The following code examples show how to use Amazon Redshift with an AWS software development kit (SDK).

#### Code examples

- [Actions for Amazon Redshift using AWS SDKs \(p. 1509\)](#)
  - Create an Amazon Redshift cluster using an AWS SDK (p. 1509)
  - Delete an Amazon Redshift cluster using an AWS SDK (p. 1512)
  - Describe an Amazon Redshift cluster using an AWS SDK (p. 1513)
  - Modify an Amazon Redshift cluster using an AWS SDK (p. 1515)
- [Cross-service examples for Amazon Redshift using AWS SDKs \(p. 1517\)](#)
  - Create an Amazon Redshift item tracker (p. 1517)

## Actions for Amazon Redshift using AWS SDKs

The following code examples show how to use Amazon Redshift with AWS SDKs. Each example calls an individual service function.

#### Examples

- [Create an Amazon Redshift cluster using an AWS SDK \(p. 1509\)](#)
- [Delete an Amazon Redshift cluster using an AWS SDK \(p. 1512\)](#)
- [Describe an Amazon Redshift cluster using an AWS SDK \(p. 1513\)](#)
- [Modify an Amazon Redshift cluster using an AWS SDK \(p. 1515\)](#)

## Create an Amazon Redshift cluster using an AWS SDK

The following code examples show how to create an Amazon Redshift cluster.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the cluster.

```
public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername, String masterUserPassword ) {

    try {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername) // set the user name here
            .masterUserPassword(masterUserPassword) // set the user password
here
            .nodeType("ds2.xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        CreateClusterResponse clusterResponse =
redshiftClient.createCluster(clusterRequest);
        System.out.println("Created cluster " +
clusterResponse.cluster().clusterIdentifier());

    } catch (RedshiftException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateCluster](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { RedshiftClient } = require("@aws-sdk/client-redshift");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Create the cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateClusterCommand } from "@aws-sdk/client-redshift";
```

```

import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
    ClusterIdentifier: "CLUSTER_NAME", // Required
    NodeType: "NODE_TYPE", //Required
    MasterUsername: "MASTER_USER_NAME", // Required - must be lowercase
    MasterUserPassword: "MASTER_USER_PASSWORD", // Required - must contain at least
    one uppercase letter, and one number
    ClusterType: "CLUSTER_TYPE", // Required
    IAMRoleARN: "IAM_ROLE_ARN", // Optional - the ARN of an IAM role with permissions
    your cluster needs to access other AWS services on your behalf, such as Amazon S3.
    ClusterSubnetGroupName: "CLUSTER_SUBNET_GROUPNAME", //Optional - the name of a
    cluster subnet group to be associated with this cluster. Defaults to 'default' if
    not specified.
    DBName: "DATABASE_NAME", // Optional - defaults to 'dev' if not specified
    Port: "PORT_NUMBER", // Optional - defaults to '5439' if not specified
};

const run = async () => {
    try {
        const data = await redshiftClient.send(new CreateClusterCommand(params));
        console.log(
            "Cluster " + data.Cluster.ClusterIdentifier + " successfully created"
        );
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();

```

- For API details, see [CreateCluster](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the cluster.

```

suspend fun createCluster(clusterId: String?, masterUsernameVal: String?,
    masterUserPasswordVal: String?) {
    val clusterRequest = CreateClusterRequest {
        clusterIdentifier = clusterId
        masterUsername = masterUsernameVal // set the user name here
        masterUserPassword = masterUserPasswordVal // set the user password here
        nodeType = "ds2.xlarge"
        publiclyAccessible = true
        numberofNodes = 2
    }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.createCluster(clusterRequest)
        println("Created cluster ${clusterResponse.cluster?.clusterIdentifier}")
    }
}

```

```
}
```

- For API details, see [CreateCluster](#) in *AWS SDK for Kotlin API reference*.

## Delete an Amazon Redshift cluster using an AWS SDK

The following code examples show how to delete an Amazon Redshift cluster.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete the cluster.

```
public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
clusterId) {

    try {
        DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .skipFinalClusterSnapshot(true)
            .build();

        DeleteClusterResponse response =
redshiftClient.deleteCluster(deleteClusterRequest);
        System.out.println("The status is
"+response.cluster().clusterStatus());

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteCluster](#) in *AWS SDK for Java 2.x API Reference*.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { RedshiftClient } = require("@aws-sdk/client-redshift");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Create the cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { DeleteClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
  ClusterIdentifier: "CLUSTER_NAME",
  SkipFinalClusterSnapshot: false,
  FinalClusterSnapshotIdentifier: "CLUSTER_SNAPSHOT_ID",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new DeleteClusterCommand(params));
    console.log("Success, cluster deleted. ", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [DeleteCluster](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete the cluster.

```
suspend fun deleteRedshiftCluster(clusterId: String?) {

    val request = DeleteClusterRequest {
        clusterIdentifier = clusterId
        skipFinalClusterSnapshot = true
    }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val response = redshiftClient.deleteCluster(request)
        println("The status is ${response.cluster?.clusterStatus}")
    }
}
```

- For API details, see [DeleteCluster](#) in *AWS SDK for Kotlin API reference*.

## Describe an Amazon Redshift cluster using an AWS SDK

The following code examples show how to describe your Amazon Redshift clusters.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Describe the cluster.

```
public static void describeRedshiftClusters(RedshiftClient redshiftClient) {  
  
    try {  
        DescribeClustersResponse clusterResponse =  
redshiftClient.describeClusters();  
        List<Cluster> clusterList = clusterResponse.clusters();  
        for (Cluster cluster: clusterList) {  
            System.out.println("Cluster database name is: "+cluster.dbName());  
            System.out.println("Cluster status is: "+cluster.clusterStatus());  
        }  
  
    } catch (RedshiftException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DescribeClusters](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { RedshiftClient } = require("@aws-sdk/client-redshift");  
// Set the AWS Region.  
const REGION = "REGION";  
//Set the Redshift Service Object  
const redshiftClient = new RedshiftClient({ region: REGION });  
export { redshiftClient };
```

Describe your clusters.

```
// Import required AWS SDK clients and commands for Node.js  
import { DescribeClustersCommand } from "@aws-sdk/client-redshift";  
import { redshiftClient } from "./libs/redshiftClient.js";  
  
const params = {  
    ClusterIdentifier: "CLUSTER_NAME",  
};  
  
const run = async () => {  
    try {  
        const data = await redshiftClient.send(new DescribeClustersCommand(params));  
    }
```

```
        console.log("Success", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
```

```
run();
```

- For API details, see [DescribeClusters](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Describe the cluster.

```
suspend fun describeRedshiftClusters() {

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse =
            redshiftClient.describeClusters(DescribeClustersRequest {})
        val clusterList = clusterResponse.clusters

        if (clusterList != null) {
            for (cluster in clusterList) {
                println("Cluster database name is ${cluster.dbName}")
                println("Cluster status is ${cluster.clusterStatus}")
            }
        }
    }
}
```

- For API details, see [DescribeClusters](#) in *AWS SDK for Kotlin API reference*.

## Modify an Amazon Redshift cluster using an AWS SDK

The following code examples show how to modify an Amazon Redshift cluster.

### Java

#### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Modify a cluster.

```
public static void modifyCluster(RedshiftClient redshiftClient, String
clusterId) {
```

```
try {
    ModifyClusterRequest modifyClusterRequest =
    ModifyClusterRequest.builder()
        .clusterIdentifier(clusterId)
        .preferredMaintenanceWindow("wed:07:30-wed:08:00")
        .build();

    ModifyClusterResponse clusterResponse =
    redshiftClient.modifyCluster(modifyClusterRequest);
    System.out.println("The modified cluster was successfully modified
and has "+ clusterResponse.cluster().preferredMaintenanceWindow() +" as the
maintenance window");

} catch (RedshiftException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [ModifyCluster](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { RedshiftClient } = require("@aws-sdk/client-redshift");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Modify a cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { ModifyClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

// Set the parameters
const params = {
  ClusterIdentifier: "CLUSTER_NAME",
  MasterUserPassword: "NEW_MASTER_USER_PASSWORD",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new ModifyClusterCommand(params));
    console.log("Success was modified.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
```

```
run();
```

- For API details, see [ModifyCluster](#) in *AWS SDK for JavaScript API Reference*.

Kotlin

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Modify a cluster.

```
suspend fun modifyCluster(clusterId: String?) {  
    val modifyClusterRequest = ModifyClusterRequest {  
        clusterIdentifier = clusterId  
        preferredMaintenanceWindow = "wed:07:30-wed:08:00"  
    }  
  
    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->  
        val clusterResponse = redshiftClient.modifyCluster(modifyClusterRequest)  
        println("The modified cluster was successfully modified and has  
        ${clusterResponse.cluster?.preferredMaintenanceWindow} as the maintenance window")  
    }  
}
```

- For API details, see [ModifyCluster](#) in *AWS SDK for Kotlin API reference*.

## Cross-service examples for Amazon Redshift using AWS SDKs

The following code examples show how to use Amazon Redshift with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

#### Examples

- [Create an Amazon Redshift item tracker \(p. 1517\)](#)

## Create an Amazon Redshift item tracker

The following code examples show how to create a web application that tracks and reports on work items using an Amazon Redshift database.

Java

#### SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon Redshift database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Redshift data and for use by a React application, see the full example on [GitHub](#).

### Services used in this example

- Amazon Redshift
- Amazon SES

Kotlin

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to create a web application that tracks and reports on work items stored in an Amazon Redshift database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Redshift data and for use by a React application, see the full example on [GitHub](#).

### Services used in this example

- Amazon Redshift
- Amazon SES

## Code examples for Amazon Rekognition using AWS SDKs

The following code examples show how to use Amazon Rekognition with an AWS software development kit (SDK).

#### Code examples

- [Actions for Amazon Rekognition using AWS SDKs \(p. 1519\)](#)
  - Compare faces in an image against a reference image with Amazon Rekognition using an AWS SDK (p. 1519)
  - Create an Amazon Rekognition collection using an AWS SDK (p. 1524)
  - Delete an Amazon Rekognition collection using an AWS SDK (p. 1526)
  - Delete faces from an Amazon Rekognition collection using an AWS SDK (p. 1529)
  - Describe an Amazon Rekognition collection using an AWS SDK (p. 1532)
  - Detect faces in an image with Amazon Rekognition using an AWS SDK (p. 1535)
  - Detect labels in an image with Amazon Rekognition using an AWS SDK (p. 1541)
  - Detect moderation labels in an image with Amazon Rekognition using an AWS SDK (p. 1545)
  - Detect text in an image with Amazon Rekognition using an AWS SDK (p. 1548)
  - Get information about celebrities with Amazon Rekognition using an AWS SDK (p. 1551)
  - Index faces to an Amazon Rekognition collection using an AWS SDK (p. 1552)
  - List Amazon Rekognition collections using an AWS SDK (p. 1556)
  - List faces in an Amazon Rekognition collection using an AWS SDK (p. 1559)
  - Recognize celebrities in an image with Amazon Rekognition using an AWS SDK (p. 1562)

- Search for faces in an Amazon Rekognition collection using an AWS SDK (p. 1566)
- Search for faces in an Amazon Rekognition collection compared to a reference image using an AWS SDK (p. 1569)
- Scenarios for Amazon Rekognition using AWS SDKs (p. 1572)
  - Build an Amazon Rekognition collection and find faces in it using an AWS SDK (p. 1572)
  - Detect and display elements in images with Amazon Rekognition using an AWS SDK (p. 1579)
  - Detect information in videos using Amazon Rekognition and the AWS SDK (p. 1588)
- Cross-service examples for Amazon Rekognition using AWS SDKs (p. 1606)
  - Detect PPE in images with Amazon Rekognition using an AWS SDK (p. 1606)
  - Detect faces in an image using an AWS SDK (p. 1607)
  - Detect objects in images with Amazon Rekognition using an AWS SDK (p. 1607)
  - Detect people and objects in a video with Amazon Rekognition using an AWS SDK (p. 1610)
  - Save EXIF and other image information using an AWS SDK (p. 1611)

## Actions for Amazon Rekognition using AWS SDKs

The following code examples show how to use Amazon Rekognition with AWS SDKs. Each example calls an individual service function.

### Examples

- Compare faces in an image against a reference image with Amazon Rekognition using an AWS SDK (p. 1519)
- Create an Amazon Rekognition collection using an AWS SDK (p. 1524)
- Delete an Amazon Rekognition collection using an AWS SDK (p. 1526)
- Delete faces from an Amazon Rekognition collection using an AWS SDK (p. 1529)
- Describe an Amazon Rekognition collection using an AWS SDK (p. 1532)
- Detect faces in an image with Amazon Rekognition using an AWS SDK (p. 1535)
- Detect labels in an image with Amazon Rekognition using an AWS SDK (p. 1541)
- Detect moderation labels in an image with Amazon Rekognition using an AWS SDK (p. 1545)
- Detect text in an image with Amazon Rekognition using an AWS SDK (p. 1548)
- Get information about celebrities with Amazon Rekognition using an AWS SDK (p. 1551)
- Index faces to an Amazon Rekognition collection using an AWS SDK (p. 1552)
- List Amazon Rekognition collections using an AWS SDK (p. 1556)
- List faces in an Amazon Rekognition collection using an AWS SDK (p. 1559)
- Recognize celebrities in an image with Amazon Rekognition using an AWS SDK (p. 1562)
- Search for faces in an Amazon Rekognition collection using an AWS SDK (p. 1566)
- Search for faces in an Amazon Rekognition collection compared to a reference image using an AWS SDK (p. 1569)

## Compare faces in an image against a reference image with Amazon Rekognition using an AWS SDK

The following code examples show how to compare faces in an image against a reference image with Amazon Rekognition.

For more information, see [Comparing faces in images](#).

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to compare faces in two images.
/// The example uses the AWS SDK for .NET 3.7 and .NET Core 5.0.
/// </summary>
public class CompareFaces
{
    public static async Task Main()
    {
        float similarityThreshold = 70F;
        string sourceImage = "source.jpg";
        string targetImage = "target.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        Amazon.Rekognition.Model.Image imageSource = new
Amazon.Rekognition.Model.Image();

        try
        {
            using FileStream fs = new FileStream(sourceImage, FileMode.Open,
FileAccess.Read);
                byte[] data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageSource.Bytes = new MemoryStream(data);
        }
        catch (Exception)
        {
            Console.WriteLine($"Failed to load source image: {sourceImage}");
            return;
        }

        Amazon.Rekognition.Model.Image imageTarget = new
Amazon.Rekognition.Model.Image();

        try
        {
            using FileStream fs = new FileStream(targetImage, FileMode.Open,
FileAccess.Read);
                byte[] data = new byte[fs.Length];
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageTarget.Bytes = new MemoryStream(data);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Failed to load target image: {targetImage}");
            Console.WriteLine(ex.Message);
            return;
        }
    }
}
```

```
var compareFacesRequest = new CompareFacesRequest
{
    SourceImage = imageSource,
    TargetImage = imageTarget,
    SimilarityThreshold = similarityThreshold,
};

// Call operation
var compareFacesResponse = await
rekognitionClient.CompareFacesAsync(compareFacesRequest);

// Display results
compareFacesResponse.FaceMatches.ForEach(match =>
{
    ComparedFace face = match.Face;
    BoundingBox position = face.BoundingBox;
    Console.WriteLine($"Face at {position.Left} {position.Top} matches
with {match.Similarity}% confidence.");
});

Console.WriteLine($"Found {compareFacesResponse.UnmatchedFaces.Count}
face(s) that did not match.");
}
```

- For API details, see [CompareFaces](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void compareTwoFaces(RekognitionClient rekClient, Float
similarityThreshold, String sourceImage, String targetImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        InputStream tarStream = new FileInputStream(targetImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        SdkBytes targetBytes = SdkBytes.fromInputStream(tarStream);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        Image tarImage = Image.builder()
            .bytes(targetBytes)
            .build();

        CompareFacesRequest facesRequest = CompareFacesRequest.builder()
            .sourceImage(souImage)
            .targetImage(tarImage)
            .similarityThreshold(similarityThreshold)
            .build();

        // Compare the two images.
        CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
        List<CompareFacesMatch> faceDetails = compareFacesResult.faceMatches();
```

```
for (CompareFacesMatch match: faceDetails){
    ComparedFace face= match.face();
    BoundingBox position = face.boundingBox();
    System.out.println("Face at " + position.left().toString()
        + " " + position.top()
        + " matches with " + face.confidence().toString()
        + "% confidence.");
}

List<ComparedFace> uncompered = compareFacesResult.unmatchedFaces();
System.out.println("There was " + uncompered.size() + " face(s) that
did not match");
System.out.println("Source image rotation: " +
compareFacesResult.sourceImageOrientationCorrection());
System.out.println("target image rotation: " +
compareFacesResult.targetImageOrientationCorrection());

} catch(RekognitionException | FileNotFoundException e) {
    System.out.println("Failed to load source image " + sourceImage);
    System.exit(1);
}
}
```

- For API details, see [CompareFaces](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun compareTwoFaces(similarityThresholdVal: Float, sourceImageVal: String,
targetImageVal: String) {

    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage = Image {
        bytes = sourceBytes
    }

    val tarImage = Image {
        bytes = targetBytes
    }

    val facesRequest = CompareFacesRequest {
        sourceImage = souImage
        targetImage = tarImage
        similarityThreshold = similarityThresholdVal
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val compareFacesResult = rekClient.compareFaces(facesRequest)
    }
}
```

```
    val faceDetails = compareFacesResult.faceMatches

    if (faceDetails != null) {
        for (match: CompareFacesMatch in faceDetails) {
            val face = match.face
            val position = face?.boundingBox
            if (position != null)
                println("Face at ${position.left} ${position.top} matches with
${face.confidence} % confidence.")
        }
    }

    val uncompered = compareFacesResult.unmatchedFaces
    if (uncompered != null)
        println("There was ${uncompered.size} face(s) that did not match")

    println("Source image rotation:
${compareFacesResult.sourceImageOrientationCorrection}")
    println("target image rotation:
${compareFacesResult.targetImageOrientationCorrection}")
}
```

- For API details, see [CompareFaces](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client

    def compare_faces(self, target_image, similarity):
        """
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value greater
                           than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                reference image. The second element is the list of faces that have
                a similarity value below the specified threshold.
        """

```

```
try:
    response = self.rekognition_client.compare_faces(
        SourceImage=self.image,
        TargetImage=target_image.image,
        SimilarityThreshold=similarity)
    matches = [RekognitionFace(match['Face']) for match
               in response['FaceMatches']]
    unmatches = [RekognitionFace(face) for face in
    response['UnmatchedFaces']]
    logger.info(
        "Found %s matched faces and %s unmatched faces.",
        len(matches), len(unmatches))
except ClientError:
    logger.exception(
        "Couldn't match faces from %s to %s.", self.image_name,
        target_image.image_name)
    raise
else:
    return matches, unmatches
```

- For API details, see [CompareFaces](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an Amazon Rekognition collection using an AWS SDK

The following code examples show how to create an Amazon Rekognition collection.

For more information, see [Creating a collection](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

///<summary>
/// Uses Amazon Rekognition to create a collection to which you can add
/// faces using the IndexFaces operation. The example was created using
/// the AWS SDK for .NET 3.7 and .NET Core 5.0.
///</summary>
public class CreateCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine("Creating collection: " + collectionId);

        var createCollectionRequest = new CreateCollectionRequest
        {
            CollectionId = collectionId,
        };
    }
}
```

```
        CreateCollectionResponse createCollectionResponse = await
rekognitionClient.CreateCollectionAsync(createCollectionRequest);
        Console.WriteLine($"CollectionArn :
{createCollectionResponse.CollectionArn}");
        Console.WriteLine($"Status code :
{createCollectionResponse.StatusCode}");
    }
}
```

- For API details, see [CreateCollection](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createMyCollection(RekognitionClient rekClient, String
collectionId ) {

    try {
        CreateCollectionRequest collectionRequest =
CreateCollectionRequest.builder()
            .collectionId(collectionId)
            .build();

        CreateCollectionResponse collectionResponse =
rekClient.createCollection(collectionRequest);
        System.out.println("CollectionArn: " +
collectionResponse.collectionArn());
        System.out.println("Status code: " +
collectionResponse.statusCode().toString());

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateCollection](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createMyCollection(collectionIdVal: String) {
```

```
    val request = CreateCollectionRequest {
        collectionId = collectionIdVal
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- For API details, see [CreateCollection](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client

    def create_collection(self, collection_id):
        """
        Creates an empty collection.

        :param collection_id: Text that identifies the collection.
        :return: The newly created collection.
        """
        try:
            response = self.rekognition_client.create_collection(
                CollectionId=collection_id)
            response['CollectionId'] = collection_id
            collection = RekognitionCollection(response, self.rekognition_client)
            logger.info("Created collection %s.", collection_id)
        except ClientError:
            logger.exception("Couldn't create collection %s.", collection_id)
            raise
        else:
            return collection
```

- For API details, see [CreateCollection](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an Amazon Rekognition collection using an AWS SDK

The following code examples show how to delete an Amazon Rekognition collection.

For more information, see [Deleting a collection](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

///<summary>
/// Uses the Amazon Rekognition Service to delete an existing collection.
/// The example was created using the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
///</summary>
public class DeleteCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine("Deleting collection: " + collectionId);

        var deleteCollectionRequest = new DeleteCollectionRequest()
        {
            CollectionId = collectionId,
        };

        var deleteCollectionResponse = await
rekognitionClient.DeleteCollectionAsync(deleteCollectionRequest);
        Console.WriteLine($"{collectionId}:
{deleteCollectionResponse.StatusCode}");
    }
}
```

- For API details, see [DeleteCollection](#) in *AWS SDK for .NET API Reference*.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteMyCollection(RekognitionClient rekClient, String
collectionId) {

    try {
        DeleteCollectionRequest deleteCollectionRequest =
DeleteCollectionRequest.builder()
            .collectionId(collectionId)
            .build();
    }
```

```
        DeleteCollectionResponse deleteCollectionResponse =
rekClient.deleteCollection(deleteCollectionRequest);
        System.out.println(collectionId + ": " +
deleteCollectionResponse.statusCode().toString());

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteCollection](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteMyCollection(collectionIdVal: String) {

    val request = DeleteCollectionRequest {
        collectionId = collectionIdVal
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- For API details, see [DeleteCollection](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.
    
```

```
:param collection: Collection data in the format returned by a call to
                  create_collection.
:param rekognition_client: A Boto3 Rekognition client.
"""
    self.collection_id = collection['CollectionId']
    self.collection_arn, self.face_count, self.created =
self._unpack_collection(
    collection)
    self.rekognition_client = rekognition_client

@staticmethod
def _unpack_collection(collection):
"""
Unpacks optional parts of a collection that can be returned by
describe_collection.

:param collection: The collection data.
:return: A tuple of the data in the collection.
"""
    return (
        collection.get('CollectionArn'),
        collection.get('FaceCount', 0),
        collection.get('CreationTimestamp'))

def delete_collection(self):
"""
Deletes the collection.
"""
    try:
        self.rekognition_client.delete_collection(CollectionId=self.collection_id)
        logger.info("Deleted collection %s.", self.collection_id)
        self.collection_id = None
    except ClientError:
        logger.exception("Couldn't delete collection %s.", self.collection_id)
        raise
```

- For API details, see [DeleteCollection](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete faces from an Amazon Rekognition collection using an AWS SDK

The following code examples show how to delete faces from an Amazon Rekognition collection.

For more information, see [Deleting faces from a collection](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
```

```
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to delete one or more faces from
/// a Rekognition collection. This example was created using the AWS SDK
/// for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DeleteFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        var faces = new List<string> { "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx" };

        var rekognitionClient = new AmazonRekognitionClient();

        var deleteFacesRequest = new DeleteFacesRequest()
        {
            CollectionId = collectionId,
            FaceIds = faces,
        };

        DeleteFacesResponse deleteFacesResponse = await
rekognitionClient.DeleteFacesAsync(deleteFacesRequest);
        deleteFacesResponse.DeletedFaces.ForEach(face =>
        {
            Console.WriteLine($"FaceID: {face}");
        });
    }
}
```

- For API details, see [DeleteFaces in AWS SDK for .NET API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteFacesCollection(RekognitionClient rekClient,
                                         String collectionId,
                                         String faceId) {

    try {
        DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
            .collectionId(collectionId)
            .faceIds(faceId)
            .build();

        rekClient.deleteFaces(deleteFacesRequest);
        System.out.println("The face was deleted from the collection.");

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteFaces in AWS SDK for Java 2.x API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteFacesCollection(collectionIdVal: String?, faceIdVal: String) {  
  
    val deleteFacesRequest = DeleteFacesRequest {  
        collectionId = collectionIdVal  
        faceIds = listOf(faceIdVal)  
    }  
  
    RekognitionClient { region = "us-east-1" }.use { rekClient ->  
        rekClient.deleteFaces(deleteFacesRequest)  
        println("$faceIdVal was deleted from the collection")  
    }  
}
```

- For API details, see [DeleteFaces in AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollection:  
    """  
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper  
    around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, collection, rekognition_client):  
        """  
        Initializes a collection object.  
  
        :param collection: Collection data in the format returned by a call to  
            create_collection.  
        :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.collection_id = collection['CollectionId']  
        self.collection_arn, self.face_count, self.created =  
        self._unpack_collection(  
            collection)  
        self.rekognition_client = rekognition_client
```

```
@staticmethod
def _unpack_collection(collection):
    """
    Unpacks optional parts of a collection that can be returned by
    describe_collection.

    :param collection: The collection data.
    :return: A tuple of the data in the collection.
    """
    return (
        collection.get('CollectionArn'),
        collection.get('FaceCount', 0),
        collection.get('CreationTimestamp'))

def delete_faces(self, face_ids):
    """
    Deletes faces from the collection.

    :param face_ids: The list of IDs of faces to delete.
    :return: The list of IDs of faces that were deleted.
    """
    try:
        response = self.rekognition_client.delete_faces(
            CollectionId=self.collection_id, FaceIds=face_ids)
        deleted_ids = response['DeletedFaces']
        logger.info(
            "Deleted %s faces from %s.", len(deleted_ids), self.collection_id)
    except ClientError:
        logger.exception("Couldn't delete faces from %s.", self.collection_id)
        raise
    else:
        return deleted_ids
```

- For API details, see [DeleteFaces in AWS SDK for Python \(Boto3\) API Reference](#).

## Describe an Amazon Rekognition collection using an AWS SDK

The following code examples show how to describe an Amazon Rekognition collection.

For more information, see [Describing a collection](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to describe the contents of a
/// collection. The example was created using the AWS SDK for .NET version
/// 3.7 and .NET Core 5.0.
/// </summary>
```

```
public class DescribeCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine($"Describing collection: {collectionId}");

        var describeCollectionRequest = new DescribeCollectionRequest()
        {
            CollectionId = collectionId,
        };

        var describeCollectionResponse = await
rekognitionClient.DescribeCollectionAsync(describeCollectionRequest);
        Console.WriteLine($"Collection ARN:
{describeCollectionResponse.CollectionARN}");
        Console.WriteLine($"Face count:
{describeCollectionResponse.FaceCount}");
        Console.WriteLine($"Face model version:
{describeCollectionResponse.FaceModelVersion}");
        Console.WriteLine($"Created:
{describeCollectionResponse.CreationTimestamp}");
    }
}
```

- For API details, see [DescribeCollection](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeColl(RekognitionClient rekClient, String
collectionName) {

    try {
        DescribeCollectionRequest describeCollectionRequest =
DescribeCollectionRequest.builder()
            .collectionId(collectionName)
            .build();

        DescribeCollectionResponse describeCollectionResponse =
rekClient.describeCollection(describeCollectionRequest);
        System.out.println("Collection Arn : " +
describeCollectionResponse.collectionARN());
        System.out.println("Created : " +
describeCollectionResponse.creationTimestamp().toString());

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeCollection](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeColl(collectionName: String) {  
  
    val request = DescribeCollectionRequest {  
        collectionId = collectionName  
    }  
  
    RekognitionClient { region = "us-east-1" }.use { rekClient ->  
        val response = rekClient.describeCollection(request)  
        println("The collection Arn is ${response.collectionArn}")  
        println("The collection contains this many faces ${response.faceCount}")  
    }  
}
```

- For API details, see [DescribeCollection](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollection:  
    """  
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper  
    around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, collection, rekognition_client):  
        """  
        Initializes a collection object.  
  
        :param collection: Collection data in the format returned by a call to  
                           create_collection.  
        :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.collection_id = collection['CollectionId']  
        self.collection_arn, self.face_count, self.created =  
        self._unpack_collection(  
            collection)  
        self.rekognition_client = rekognition_client  
  
    @staticmethod  
    def _unpack_collection(collection):  
        """  
        Unpacks optional parts of a collection that can be returned by  
        describe_collection.  
    """
```

```
:param collection: The collection data.  
:return: A tuple of the data in the collection.  
"""  
    return (  
        collection.get('CollectionArn'),  
        collection.get('FaceCount', 0),  
        collection.get('CreationTimestamp'))  
  
def describe_collection(self):  
    """  
        Gets data about the collection from the Amazon Rekognition service.  
        :return: The collection rendered as a dict.  
    """  
    try:  
        response = self.rekognition_client.describe_collection(  
            CollectionId=self.collection_id)  
        # Work around capitalization of Arn vs. ARN  
        response['CollectionArn'] = response.get('CollectionARN')  
        (self.collection_arn, self.face_count,  
         self.created) = self._unpack_collection(response)  
        logger.info("Got data for collection %s.", self.collection_id)  
    except ClientError:  
        logger.exception("Couldn't get data for collection %s.",  
                         self.collection_id)  
        raise  
    else:  
        return self.to_dict()
```

- For API details, see [DescribeCollection](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect faces in an image with Amazon Rekognition using an AWS SDK

The following code examples show how to detect faces in an image with Amazon Rekognition.

For more information, see [Detecting faces in an image](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Collections.Generic;  
using System.Threading.Tasks;  
using Amazon.Rekognition;  
using Amazon.Rekognition.Model;  
  
/// <summary>  
/// Uses the Amazon Rekognition Service to detect faces within an image  
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket. This  
/// example uses the AWS SDK for .NET version 3.7 and .NET Core 5.0.  
/// </summary>  
public class DetectFaces
```

```
{
    public static async Task Main()
    {
        string photo = "input.jpg";
        string bucket = "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectFacesRequest = new DetectFacesRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },

            // Attributes can be "ALL" or "DEFAULT".
            // "DEFAULT": BoundingBox, Confidence, Landmarks, Pose, and
            Quality.
            // "ALL": See https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Rekognition/TFaceDetail.html
            Attributes = new List<string>() { "ALL" },
        };

        try
        {
            DetectFacesResponse detectFacesResponse = await
rekognitionClient.DetectFacesAsync(detectFacesRequest);
            bool hasAll = detectFacesRequest.Attributes.Contains("ALL");
            foreach (FaceDetail face in detectFacesResponse.FaceDetails)
            {
                Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
left={face.BoundingBox.Top} width={face.BoundingBox.Width}
height={face.BoundingBox.Height}");
                Console.WriteLine($"Confidence: {face.Confidence}");
                Console.WriteLine($"Landmarks: {face.Landmarks.Count}");
                Console.WriteLine($"Pose: pitch={face.Pose.Pitch}
roll={face.Pose.Roll} yaw={face.Pose.Yaw}");
                Console.WriteLine($"Brightness:
{face.Quality.Brightness}\tSharpness: {face.Quality.Sharpness}");

                if (hasAll)
                {
                    Console.WriteLine($"Estimated age is between
{face.AgeRange.Low} and {face.AgeRange.High} years old.");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
}
```

Display bounding box information for all faces in an image.

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
```

```
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

///<summary>
/// Uses the Amazon Rekognition Service to display the details of the
/// bounding boxes around the faces detected in an image. This example was
/// created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
///</summary>
public class ImageOrientationBoundingBox
{
    public static async Task Main()
    {
        string photo = @"D:\Development\AWS-Examples\Rekognition
\target.jpg"; // "photo.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        var image = new Amazon.Rekognition.Model.Image();
        try
        {
            using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
            byte[] data = null;
            data = new byte[fs.Length];
            fs.Read(data, 0, (int)fs.Length);
            image.Bytes = new MemoryStream(data);
        }
        catch (Exception)
        {
            Console.WriteLine("Failed to load file " + photo);
            return;
        }

        int height;
        int width;

        // Used to extract original photo width/height
        using (var imageBitmap = new Bitmap(photo))
        {
            height = imageBitmap.Height;
            width = imageBitmap.Width;
        }

        Console.WriteLine("Image Information:");
        Console.WriteLine(photo);
        Console.WriteLine("Image Height: " + height);
        Console.WriteLine("Image Width: " + width);

        try
        {
            var detectFacesRequest = new DetectFacesRequest()
            {
                Image = image,
                Attributes = new List<string>() { "ALL" },
            };

            DetectFacesResponse detectFacesResponse = await
rekognitionClient.DetectFacesAsync(detectFacesRequest);
            detectFacesResponse.FaceDetails.ForEach(face =>
            {
                Console.WriteLine("Face:");
                ShowBoundingBoxPositions(
                    height,
                    width,
                    face.BoundingBox,
```

```

        detectFacesResponse.OrientationCorrection);

        Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
left={face.BoundingBox.Top} width={face.BoundingBox.Width}
height={face.BoundingBox.Height}");
        Console.WriteLine($"The detected face is estimated to be
between {face.AgeRange.Low} and {face.AgeRange.High} years old.\n");
    });
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
}

/// <summary>
/// Display the bounding box information for an image.
/// </summary>
/// <param name="imageHeight">The height of the image.</param>
/// <param name="imageWidth">The width of the image.</param>
/// <param name="box">The bounding box for a face found within the image.</param>
/// <param name="rotation">The rotation of the face's bounding box.</param>
public static void ShowBoundingBoxPositions(int imageHeight, int
imageWidth, BoundingBox box, string rotation)
{
    float left;
    float top;

    if (rotation == null)
    {
        Console.WriteLine("No estimated orientation. Check Exif data.");
        return;
    }

    // Calculate face position based on image orientation.
    switch (rotation)
    {
        case "ROTATE_0":
            left = imageWidth * box.Left;
            top = imageHeight * box.Top;
            break;
        case "ROTATE_90":
            left = imageHeight * (1 - (box.Top + box.Height));
            top = imageWidth * box.Left;
            break;
        case "ROTATE_180":
            left = imageWidth - (imageWidth * (box.Left + box.Width));
            top = imageHeight * (1 - (box.Top + box.Height));
            break;
        case "ROTATE_270":
            left = imageHeight * box.Top;
            top = imageWidth * (1 - box.Left - box.Width);
            break;
        default:
            Console.WriteLine("No estimated orientation information. Check
Exif data.");
            return;
    }

    // Display face location information.
    Console.WriteLine($"Left: {left}");
    Console.WriteLine($"Top: {top}");
    Console.WriteLine($"Face Width: {imageWidth * box.Width}");
    Console.WriteLine($"Face Height: {imageHeight * box.Height}");
}

```

```
}
```

- For API details, see [DetectFaces in AWS SDK for .NET API Reference](#).

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectFacesinImage(RekognitionClient rekClient, String sourceImage) {  
  
    try {  
        InputStream sourceStream = new FileInputStream(sourceImage);  
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);  
  
        // Create an Image object for the source image.  
        Image souImage = Image.builder()  
            .bytes(sourceBytes)  
            .build();  
  
        DetectFacesRequest facesRequest = DetectFacesRequest.builder()  
            .attributes(Attribute.ALL)  
            .image(souImage)  
            .build();  
  
        DetectFacesResponse facesResponse =  
rekClient.detectFaces(facesRequest);  
        List<FaceDetail> faceDetails = facesResponse.faceDetails();  
        for (FaceDetail face : faceDetails) {  
            AgeRange ageRange = face.ageRange();  
            System.out.println("The detected face is estimated to be between "  
                + ageRange.low().toString() + " and " +  
ageRange.high().toString()  
                + " years old.");  
  
            System.out.println("There is a smile :  
"+face.smile().value().toString());  
        }  
  
    } catch (RekognitionException | FileNotFoundException e) {  
        System.out.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DetectFaces in AWS SDK for Java 2.x API Reference](#).

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun detectFacesinImage(sourceImage: String?) {  
  
    val souImage = Image {  
        bytes = (File(sourceImage).readBytes())  
    }  
  
    val request = DetectFacesRequest {  
        attributes = listOf(Attribute.All)  
        image = souImage  
    }  
  
    RekognitionClient { region = "us-east-1" }.use { rekClient ->  
        val response = rekClient.detectFaces(request)  
        response.faceDetails?.forEach { face ->  
            val ageRange = face.ageRange  
            println("The detected face is estimated to be between ${ageRange?.low}  
and ${ageRange?.high} years old.")  
            println("There is a smile ${face.smile?.value}")  
        }  
    }  
}
```

- For API details, see [DetectFaces in AWS SDK for Kotlin API reference](#).

Python

**SDK for Python (Boto3)**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionImage:  
    """  
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper  
    around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, image, image_name, rekognition_client):  
        """  
        Initializes the image object.  
  
        :param image: Data that defines the image, either the image bytes or  
                     an Amazon S3 bucket and object key.  
        :param image_name: The name of the image.  
        :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.image = image  
        self.image_name = image_name  
        self.rekognition_client = rekognition_client  
  
    def detect_faces(self):  
        """  
        Detects faces in the image.  
  
        :return: The list of faces found in the image.  
    """
```

```
"""
try:
    response = self.rekognition_client.detect_faces(
        Image=self.image, Attributes=['ALL'])
    faces = [RekognitionFace(face) for face in response['FaceDetails']]
    logger.info("Detected %s faces.", len(faces))
except ClientError:
    logger.exception("Couldn't detect faces in %s.", self.image_name)
    raise
else:
    return faces
```

- For API details, see [DetectFaces in AWS SDK for Python \(Boto3\) API Reference](#).

## Detect labels in an image with Amazon Rekognition using an AWS SDK

The following code examples show how to detect labels in an image with Amazon Rekognition.

For more information, see [Detecting labels in an image](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect labels within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket. This
/// example was created using the AWS SDK for .NET and .NET Core 5.0.
/// </summary>
public class DetectLabels
{
    public static async Task Main()
    {
        string photo = "del_river_02092020_01.jpg"; // "input.jpg";
        string bucket = "igsmiths3photos"; // "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectlabelsRequest = new DetectLabelsRequest
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },
            MaxLabels = 10,
            MinConfidence = 75F,
```

```
};

try
{
    DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
    Console.WriteLine("Detected labels for " + photo);
    foreach (Label label in detectLabelsResponse.Labels)
    {
        Console.WriteLine($"Name: {label.Name} Confidence:
{label.Confidence}");
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
}
```

Detect labels in an image file stored on your computer.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect labels within an image
/// stored locally. This example was created using the AWS SDK for .NET
/// and .NET Core 5.0.
/// </summary>
public class DetectLabelsLocalFile
{
    public static async Task Main()
    {
        string photo = "input.jpg";

        var image = new Amazon.Rekognition.Model.Image();
        try
        {
            using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
            byte[] data = null;
            data = new byte[fs.Length];
            fs.Read(data, 0, (int)fs.Length);
            image.Bytes = new MemoryStream(data);
        }
        catch (Exception)
        {
            Console.WriteLine("Failed to load file " + photo);
            return;
        }

        var rekognitionClient = new AmazonRekognitionClient();

        var detectlabelsRequest = new DetectLabelsRequest
        {
            Image = image,
            MaxLabels = 10,
            MinConfidence = 77F,
        };
    }
}
```

```
        try
    {
        DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
        Console.WriteLine($"Detected labels for {photo}");
        foreach (Label label in detectLabelsResponse.Labels)
        {
            Console.WriteLine($"{label.Name}: {label.Confidence}");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

- For API details, see [DetectLabels](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectImageLabels(RekognitionClient rekClient, String
sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
            .image(souImage)
            .maxLabels(10)
            .build();

        DetectLabelsResponse labelsResponse =
rekClient.detectLabels(detectLabelsRequest);
        List<Label> labels = labelsResponse.labels();
        System.out.println("Detected labels for the given photo");
        for (Label label: labels) {
            System.out.println(label.name() + ": " +
label.confidence().toString());
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DetectLabels](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun detectImageLabels(sourceImage: String) {  
  
    val souImage = Image {  
        bytes = (File(sourceImage).readBytes())  
    }  
    val request = DetectLabelsRequest {  
        image = souImage  
        maxLabels = 10  
    }  
  
    RekognitionClient { region = "us-east-1" }.use { rekClient ->  
        val response = rekClient.detectLabels(request)  
        response.labels?.forEach { label ->  
            println("${label.name} : ${label.confidence}")  
        }  
    }  
}
```

- For API details, see [DetectLabels](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionImage:  
    """  
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper  
    around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, image, image_name, rekognition_client):  
        """  
        Initializes the image object.  
  
        :param image: Data that defines the image, either the image bytes or  
                     an Amazon S3 bucket and object key.  
        :param image_name: The name of the image.  
        :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.image = image  
        self.image_name = image_name  
        self.rekognition_client = rekognition_client  
  
    def detect_labels(self, max_labels):
```

```
"""
    Detects labels in the image. Labels are objects and people.

:param max_labels: The maximum number of labels to return.
:return: The list of labels detected in the image.
"""
try:
    response = self.rekognition_client.detect_labels(
        Image=self.image, MaxLabels=max_labels)
    labels = [RekognitionLabel(label) for label in response['Labels']]
    logger.info("Found %s labels in %s.", len(labels), self.image_name)
except ClientError:
    logger.info("Couldn't detect labels in %s.", self.image_name)
    raise
else:
    return labels
```

- For API details, see [DetectLabels](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect moderation labels in an image with Amazon Rekognition using an AWS SDK

The following code examples show how to detect moderation labels in an image with Amazon Rekognition. Moderation labels identify content that may be inappropriate for some audiences.

For more information, see [Detecting inappropriate images](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect unsafe content in a
/// JPEG or PNG format image. This example was created using the AWS SDK
/// for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DetectModerationLabels
{
    public static async Task Main(string[] args)
    {
        string photo = "input.jpg";
        string bucket = "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectModerationLabelsRequest = new DetectModerationLabelsRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
```

```
        {
            Name = photo,
            Bucket = bucket,
        },
    ],
    MinConfidence = 60F,
};

try
{
    var detectModerationLabelsResponse = await
rekognitionClient.DetectModerationLabelsAsync(detectModerationLabelsRequest);
    Console.WriteLine("Detected labels for " + photo);
    foreach (ModerationLabel label in
detectModerationLabelsResponse.ModerationLabels)
    {
        Console.WriteLine($"Label: {label.Name}");
        Console.WriteLine($"Confidence: {label.Confidence}");
        Console.WriteLine($"Parent: {label.ParentName}");
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
}
}
```

- For API details, see [DetectModerationLabels](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectModLabels(RekognitionClient rekClient, String
sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
            .image(souImage)
            .minConfidence(60F)
            .build();

        DetectModerationLabelsResponse moderationLabelsResponse =
rekClient.detectModerationLabels(moderationLabelsRequest);
        List<ModerationLabel> labels =
moderationLabelsResponse.moderationLabels();
        System.out.println("Detected labels for image");

        for (ModerationLabel label : labels) {
            System.out.println("Label: " + label.name())
    }
}
```

```
+ "\n Confidence: " + label.confidence().toString() + "%"
+ "\n Parent:" + label.parentName());
}

} catch (RekognitionException | FileNotFoundException e) {
    e.printStackTrace();
    System.exit(1);
}
}
```

- For API details, see [DetectModerationLabels](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun detectModLabels(sourceImage: String) {

    val myImage = Image {
        this.bytes = (File(sourceImage).readBytes())
    }

    val request = DetectModerationLabelsRequest {
        image = myImage
        minConfidence = 60f
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} %
Parent: ${label.parentName}")
        }
    }
}
```

- For API details, see [DetectModerationLabels](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
```

```
around parts of the Boto3 Amazon Rekognition API.  
"""  
def __init__(self, image, image_name, rekognition_client):  
    """  
    Initializes the image object.  
  
    :param image: Data that defines the image, either the image bytes or  
        an Amazon S3 bucket and object key.  
    :param image_name: The name of the image.  
    :param rekognition_client: A Boto3 Rekognition client.  
    """  
    self.image = image  
    self.image_name = image_name  
    self.rekognition_client = rekognition_client  
  
def detect_moderation_labels(self):  
    """  
    Detects moderation labels in the image. Moderation labels identify content  
    that may be inappropriate for some audiences.  
  
    :return: The list of moderation labels found in the image.  
    """  
    try:  
        response = self.rekognition_client.detect_moderation_labels(  
            Image=self.image)  
        labels = [RekognitionModerationLabel(label)  
            for label in response['ModerationLabels']]  
        logger.info(  
            "Found %s moderation labels in %s.", len(labels), self.image_name)  
    except ClientError:  
        logger.exception(  
            "Couldn't detect moderation labels in %s.", self.image_name)  
        raise  
    else:  
        return labels
```

- For API details, see [DetectModerationLabels](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect text in an image with Amazon Rekognition using an AWS SDK

The following code examples show how to detect text in an image with Amazon Rekognition.

For more information, see [Detecting text in an image](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Threading.Tasks;  
using Amazon.Rekognition;  
using Amazon.Rekognition.Model;  
  
/// <summary>  
/// Uses the Amazon Rekognition Service to detect text in an image. The
```

```
/// example was created using the AWS SDK for .NET version 3.7 and .NET
/// Core 5.0.
/// </summary>
public class DetectText
{
    public static async Task Main()
    {
        string photo = "Dad_photographer.jpg"; // "input.jpg";
        string bucket = "igsmiths3photos"; // "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectTextRequest = new DetectTextRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },
        };

        try
        {
            DetectTextResponse detectTextResponse = await
rekognitionClient.DetectTextAsync(detectTextRequest);
            Console.WriteLine($"Detected lines and words for {photo}");
            detectTextResponse.TextDetections.ForEach(text =>
            {
                Console.WriteLine($"Detected: {text.DetectedText}");
                Console.WriteLine($"Confidence: {text.Confidence}");
                Console.WriteLine($"Id : {text.Id}");
                Console.WriteLine($"Parent Id: {text.ParentId}");
                Console.WriteLine($"Type: {text.Type}");
            });
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```

- For API details, see [DetectText](#) in [AWS SDK for .NET API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectTextLabels(RekognitionClient rekClient, String
sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
```

```
.bytes(sourceBytes)
.build();

DetectTextRequest textRequest = DetectTextRequest.builder()
.image(souImage)
.build();

DetectTextResponse textResponse = rekClient.detectText(textRequest);
List<TextDetection> textCollection = textResponse.textDetections();
System.out.println("Detected lines and words");
for (TextDetection text: textCollection) {
    System.out.println("Detected: " + text.detectedText());
    System.out.println("Confidence: " + text.confidence().toString());
    System.out.println("Id : " + text.id());
    System.out.println("Parent Id: " + text.parentId());
    System.out.println("Type: " + text.type());
    System.out.println();
}
} catch (RekognitionException | FileNotFoundException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [DetectText](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun detectTextLabels(sourceImage: String?) {

    val souImage = Image {
        bytes = (File(sourceImage).readBytes())
    }

    val request = DetectTextRequest {
        image = souImage
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id: ${text.parentId}")
            println("Type: ${text.type}")
        }
    }
}
```

- For API details, see [DetectText](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionImage:  
    """  
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper  
    around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, image, image_name, rekognition_client):  
        """  
        Initializes the image object.  
  
        :param image: Data that defines the image, either the image bytes or  
                     an Amazon S3 bucket and object key.  
        :param image_name: The name of the image.  
        :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.image = image  
        self.image_name = image_name  
        self.rekognition_client = rekognition_client  
  
    def detect_text(self):  
        """  
        Detects text in the image.  
  
        :return The list of text elements found in the image.  
        """  
        try:  
            response = self.rekognition_client.detect_text(Image=self.image)  
            texts = [RekognitionText(text) for text in response['TextDetections']]  
            logger.info("Found %s texts in %s.", len(texts), self.image_name)  
        except ClientError:  
            logger.exception("Couldn't detect text in %s.", self.image_name)  
            raise  
        else:  
            return texts
```

- For API details, see [DetectText](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get information about celebrities with Amazon Rekognition using an AWS SDK

The following code example shows how to get information about celebrities using Amazon Rekognition.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to retrieve information about the
/// celebrity identified by the supplied celebrity Id. This example was
/// created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class CelebrityInfo
{
    public static async Task Main()
    {
        string celebId = "nnnnnnnn";

        var rekognitionClient = new AmazonRekognitionClient();

        var celebrityInfoRequest = new GetCelebrityInfoRequest
        {
            Id = celebId,
        };

        Console.WriteLine($"Getting information for celebrity: {celebId}");

        var celebrityInfoResponse = await
rekognitionClient.GetCelebrityInfoAsync(celebrityInfoRequest);

        // Display celebrity information.
        Console.WriteLine($"celebrity name: {celebrityInfoResponse.Name}");
        Console.WriteLine("Further information (if available):");
        celebrityInfoResponseUrls.ForEach(url =>
        {
            Console.WriteLine(url);
        });
    }
}
```

- For API details, see [GetCelebrityInfo](#) in *AWS SDK for .NET API Reference*.

## Index faces to an Amazon Rekognition collection using an AWS SDK

The following code examples show how to index faces in an image and add them to an Amazon Rekognition collection.

For more information, see [Adding faces to a collection](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
```

```
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

///<summary>
/// Uses the Amazon Rekognition Service to detect faces in an image
/// that has been uploaded to an Amazon Simple Storage Service (Amazon S3)
/// bucket and then adds the information to a collection. The example was
/// created using the AWS SDK for .NET and .NET Core 5.0.
///</summary>
public class AddFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection2";
        string bucket = "doc-example-bucket";
        string photo = "input.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        var image = new Image
        {
            S3Object = new S3Object
            {
                Bucket = bucket,
                Name = photo,
            },
        };

        var indexFacesRequest = new IndexFacesRequest
        {
            Image = image,
            CollectionId = collectionId,
            ExternalImageId = photo,
            DetectionAttributes = new List<string>() { "ALL" },
        };

        IndexFacesResponse indexFacesResponse = await
rekognitionClient.IndexFacesAsync(indexFacesRequest);

        Console.WriteLine($"{photo} added");
        foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
        {
            Console.WriteLine($"Face detected: Faceid is
{faceRecord.Face.FaceId}");
        }
    }
}
```

- For API details, see [IndexFaces](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void addToCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
```

```
try {
    InputStream sourceStream = new FileInputStream(sourceImage);
    SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
    Image souImage = Image.builder()
        .bytes(sourceBytes)
        .build();

    IndexFacesRequest facesRequest = IndexFacesRequest.builder()
        .collectionId(collectionId)
        .image(souImage)
        .maxFaces(1)
        .qualityFilter(QualityFilter.AUTO)
        .detectionAttributes(Attribute.DEFAULT)
        .build();

    IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
    System.out.println("Results for the image");
    System.out.println("\n Faces indexed:");
    List<FaceRecord> faceRecords = facesResponse.faceRecords();
    for (FaceRecord faceRecord : faceRecords) {
        System.out.println(" Face ID: " + faceRecord.face().faceId());
        System.out.println(" Location:" +
faceRecord.faceDetail().boundingBox().toString());
    }

    List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
    System.out.println("Faces not indexed:");
    for (UnindexedFace unindexedFace : unindexedFaces) {
        System.out.println(" Location:" +
unindexedFace.faceDetail().boundingBox().toString());
        System.out.println(" Reasons:");
        for (Reason reason : unindexedFace.reasons()) {
            System.out.println("Reason: " + reason);
        }
    }

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [IndexFaces](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun addToCollection(collectionIdVal: String?, sourceImage: String) {

    val souImage = Image {
        bytes = (File(sourceImage).readBytes())
    }
```

```

    val request = IndexFacesRequest {
        collectionId = collectionIdVal
        image = souImage
        maxFaces = 1
        qualityFilter = QualityFilter.Auto
        detectionAttributes = listOf(Attribute.Default)
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)

        // Display the results.
        println("Results for the image")
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
            println("Face ID: ${faceRecord.face?.faceId}")
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
            println("Reasons:")

            unindexedFace.reasons?.forEach { reason ->
                println("Reason: $reason")
            }
        }
    }
}

```

- For API details, see [IndexFaces](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection['CollectionId']
        self.collection_arn, self.face_count, self.created =
    self._unpack_collection(
        collection)
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):

```

```
"""
Unpacks optional parts of a collection that can be returned by
describe_collection.

:param collection: The collection data.
:return: A tuple of the data in the collection.
"""
return (
    collection.get('CollectionArn'),
    collection.get('FaceCount', 0),
    collection.get('CreationTimestamp'))

def index_faces(self, image, max_faces):
    """
    Finds faces in the specified image, indexes them, and stores them in the
    collection.

    :param image: The image to index.
    :param max_faces: The maximum number of faces to index.
    :return: A tuple. The first element is a list of indexed faces.
            The second element is a list of faces that couldn't be indexed.
    """
    try:
        response = self.rekognition_client.index_faces(
            CollectionId=self.collection_id, Image=image.image,
            ExternalImageId=image.image_name, MaxFaces=max_faces,
            DetectionAttributes=['ALL'])
        indexed_faces = [
            RekognitionFace({**face['Face'], **face['FaceDetail']})
            for face in response['FaceRecords']]
        unindexed_faces = [
            RekognitionFace(face['FaceDetail'])
            for face in response['UnindexedFaces']]
        logger.info(
            "Indexed %s faces in %s. Could not index %s faces.",
            len(indexed_faces),
            image.image_name, len(unindexed_faces))
    except ClientError:
        logger.exception("Couldn't index faces in image %s.", image.image_name)
        raise
    else:
        return indexed_faces, unindexed_faces

```

- For API details, see [IndexFaces](#) in *AWS SDK for Python (Boto3) API Reference*.

## List Amazon Rekognition collections using an AWS SDK

The following code examples show how to list Amazon Rekognition collections.

For more information, see [Listing collections](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
```

```
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

///<summary>
/// Uses the Amazpon Rekognition Service to list the collection IDs in the
/// default user account. This example was crated using the AWS SDK for
/// .NET version 3.7 and .NET Core 5.0.
///</summary>
public class ListCollections
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        Console.WriteLine("Listing collections");
        int limit = 10;

        var listCollectionsRequest = new ListCollectionsRequest
        {
            MaxResults = limit,
        };

        var listCollectionsResponse = new ListCollectionsResponse();

        do
        {
            if (listCollectionsResponse is not null)
            {
                listCollectionsRequest.NextToken =
listCollectionsResponse.NextToken;
            }

            listCollectionsResponse = await
rekognitionClient.ListCollectionsAsync(listCollectionsRequest);

            listCollectionsResponse.CollectionIds.ForEach(id =>
            {
                Console.WriteLine(id);
            });
        }
        while (listCollectionsResponse.NextToken is not null);
    }
}
```

- For API details, see [ListCollections](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllCollections(RekognitionClient rekClient) {
    try {
        ListCollectionsRequest listCollectionsRequest =
ListCollectionsRequest.builder()
        .maxResults(10)
        .build();
```

```
    ListCollectionsResponse response =
rekClient.listCollections(listCollectionsRequest);
List<String> collectionIds = response.collectionIds();
for (String resultId : collectionIds) {
    System.out.println(resultId);
}

} catch (RekognitionException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [ListCollections](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAllCollections() {

    val request = ListCollectionsRequest {
        maxResults = 10
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- For API details, see [ListCollections](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, rekognition_client):
```

```
"""
Initializes the collection manager object.

:param rekognition_client: A Boto3 Rekognition client.
"""
self.rekognition_client = rekognition_client

def list_collections(self, max_results):
    """
    Lists collections for the current account.

    :param max_results: The maximum number of collections to return.
    :return: The list of collections for the current account.
    """
    try:
        response =
self.rekognition_client.list_collections(MaxResults=max_results)
        collections = [
            RekognitionCollection({'CollectionId': col_id},
self.rekognition_client)
            for col_id in response['CollectionIds']]
    except ClientError:
        logger.exception("Couldn't list collections.")
        raise
    else:
        return collections
```

- For API details, see [ListCollections](#) in *AWS SDK for Python (Boto3) API Reference*.

## List faces in an Amazon Rekognition collection using an AWS SDK

The following code examples show how to list faces in an Amazon Rekognition collection.

For more information, see [Listing faces in a collection](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to retrieve the list of faces
/// stored in a collection. The example was created using AWS SDK for
/// .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class ListFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection2";
```

```
var rekognitionClient = new AmazonRekognitionClient();

var listFacesResponse = new ListFacesResponse();
Console.WriteLine($"Faces in collection {collectionId}");

var listFacesRequest = new ListFacesRequest
{
    CollectionId = collectionId,
    MaxResults = 1,
};

do
{
    listFacesResponse = await
rekognitionClient.ListFacesAsync(listFacesRequest);
    listFacesResponse.Faces.ForEach(face =>
    {
        Console.WriteLine(face.FaceId);
    });

    listFacesRequest.NextToken = listFacesResponse.NextToken;
}
while (!string.IsNullOrEmpty(listFacesResponse.NextToken));
}
```

- For API details, see [ListFaces](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listFacesCollection(RekognitionClient rekClient, String
collectionId ) {
    try {
        ListFacesRequest facesRequest = ListFacesRequest.builder()
            .collectionId(collectionId)
            .maxResults(10)
            .build();

        ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
        List<Face> faces = facesResponse.faces();
        for (Face face: faces) {
            System.out.println("Confidence level there is a face:
"+face.confidence());
            System.out.println("The face Id value is "+face.faceId());
        }
    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListFaces](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listFacesCollection(collectionIdVal: String?) {  
  
    val request = ListFacesRequest {  
        collectionId = collectionIdVal  
        maxResults = 10  
    }  
  
    RekognitionClient { region = "us-east-1" }.use { rekClient ->  
        val response = rekClient.listFaces(request)  
        response.faces?.forEach { face ->  
            println("Confidence level there is a face: ${face.confidence}")  
            println("The face Id value is ${face.faceId}")  
        }  
    }  
}
```

- For API details, see [ListFaces](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollection:  
    """  
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper  
    around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, collection, rekognition_client):  
        """  
        Initializes a collection object.  
  
        :param collection: Collection data in the format returned by a call to  
                          create_collection.  
        :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.collection_id = collection['CollectionId']  
        self.collection_arn, self.face_count, self.created =  
        self._unpack_collection(  
            collection)  
        self.rekognition_client = rekognition_client  
  
    @staticmethod  
    def _unpack_collection(collection):  
        """
```

```
Unpacks optional parts of a collection that can be returned by
describe_collection.

:param collection: The collection data.
:return: A tuple of the data in the collection.
"""
    return (
        collection.get('CollectionArn'),
        collection.get('FaceCount', 0),
        collection.get('CreationTimestamp'))

def list_faces(self, max_results):
    """
    Lists the faces currently indexed in the collection.

    :param max_results: The maximum number of faces to return.
    :return: The list of faces in the collection.
    """
    try:
        response = self.rekognition_client.list_faces(
            CollectionId=self.collection_id, MaxResults=max_results)
        faces = [RekognitionFace(face) for face in response['Faces']]
        logger.info(
            "Found %s faces in collection %s.", len(faces), self.collection_id)
    except ClientError:
        logger.exception(
            "Couldn't list faces in collection %s.", self.collection_id)
        raise
    else:
        return faces
```

- For API details, see [ListFaces](#) in *AWS SDK for Python (Boto3) API Reference*.

## Recognize celebrities in an image with Amazon Rekognition using an AWS SDK

The following code examples show how to recognize celebrities in an image with Amazon Rekognition.

For more information, see [Recognizing celebrities in an image](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to identify celebrities in a photo.
/// This example was created using the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
/// </summary>
public class CelebritiesInImage
```

```
{  
    public static async Task Main(string[] args)  
    {  
        string photo = "moviestars.jpg";  
  
        var rekognitionClient = new AmazonRekognitionClient();  
  
        var recognizeCelebritiesRequest = new RecognizeCelebritiesRequest();  
  
        var img = new Amazon.Rekognition.Model.Image();  
        byte[] data = null;  
        try  
        {  
            using var fs = new FileStream(photo, FileMode.Open,  
FileAccess.Read);  
            data = new byte[fs.Length];  
            fs.Read(data, 0, (int)fs.Length);  
        }  
        catch (Exception)  
        {  
            Console.WriteLine($"Failed to load file {photo}");  
            return;  
        }  
  
        img.Bytes = new MemoryStream(data);  
        recognizeCelebritiesRequest.Image = img;  
  
        Console.WriteLine($"Looking for celebrities in image {photo}\n");  
  
        var recognizeCelebritiesResponse = await  
rekognitionClient.RecognizeCelebritiesAsync(recognizeCelebritiesRequest);  
  
        Console.WriteLine($"{recognizeCelebritiesResponse.CelebrityFaces.Count}  
celebrity(s) were recognized.\n");  
        recognizeCelebritiesResponse.CelebrityFaces.ForEach(celeb =>  
        {  
            Console.WriteLine($"Celebrity recognized: {celeb.Name}");  
            Console.WriteLine($"Celebrity ID: {celeb.Id}");  
            BoundingBox boundingBox = celeb.Face.BoundingBox;  
            Console.WriteLine($"position: {boundingBox.Left}  
{boundingBox.Top}");  
            Console.WriteLine("Further information (if available):");  
            celeb.Urls.ForEach(url =>  
            {  
                Console.WriteLine(url);  
            });  
        });  
  
        Console.WriteLine($"{recognizeCelebritiesResponse.UnrecognizedFaces.Count} face(s)  
were unrecognized.");  
    }  
}
```

- For API details, see [RecognizeCelebrities](#) in *AWS SDK for .NET API Reference*.

Java

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void recognizeAllCelebrities(RekognitionClient rekClient, String sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        RecognizeCelebritiesRequest request =
        RecognizeCelebritiesRequest.builder()
            .image(souImage)
            .build();

        RecognizeCelebritiesResponse result =
        rekClient.recognizeCelebrities(request) ;
        List<Celebrity> celebs=result.celebrityFaces();
        System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
        for (Celebrity celebrity: celebs) {
            System.out.println("Celebrity recognized: " + celebrity.name());
            System.out.println("Celebrity ID: " + celebrity.id());

            System.out.println("Further information (if available):");
            for (String url: celebrity.urls()){
                System.out.println(url);
            }
            System.out.println();
        }
        System.out.println(result.unrecognizedFaces().size() + " face(s) were
unrecognized.");
    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [RecognizeCelebrities](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {

    val souImage = Image {
        bytes = (File(sourceImage).readBytes())
    }

    val request = RecognizeCelebritiesRequest {
        image = souImage
    }
}
```

```

        }

        RekognitionClient { region = "us-east-1" }.use { rekClient ->
            val response = rekClient.recognizeCelebrities(request)
            response.celebrityFaces?.forEach { celebrity ->
                println("Celebrity recognized: ${celebrity.name}")
                println("Celebrity ID:${celebrity.id}")
                println("Further information (if available):")
                celebrity.urls?.forEach { url ->
                    println(url)
                }
            }
            println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
        }
    }
}

```

- For API details, see [RecognizeCelebrities](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client

    def recognize_celebrities(self):
        """
        Detects celebrities in the image.

        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
        try:
            response = self.rekognition_client.recognize_celebrities(
                Image=self.image)
            celebrities = [RekognitionCelebrity(celeb)
                           for celeb in response['CelebrityFaces']]
            other_faces = [RekognitionFace(face)
                           for face in response['UnrecognizedFaces']]
            logger.info(
                "Found %s celebrities and %s other faces in %s.", len(celebrities),
                len(other_faces), self.image_name)
        
```

```
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.", self.image_name)
            raise
        else:
            return celebrities, other_faces
```

- For API details, see [RecognizeCelebrities](#) in *AWS SDK for Python (Boto3) API Reference*.

## Search for faces in an Amazon Rekognition collection using an AWS SDK

The following code examples show how to search for faces in an Amazon Rekognition collection that match another face from the collection.

For more information, see [Searching for a face \(face ID\)](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

///<summary>
/// Uses the Amazon Rekognition Service to find faces in an image that
/// match the face Id provided in the method request. This example was
/// created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
///</summary>
public class SearchFacesMatchingId
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        string faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";

        var rekognitionClient = new AmazonRekognitionClient();

        // Search collection for faces matching the face id.
        var searchFacesRequest = new SearchFacesRequest
        {
            CollectionId = collectionId,
            FaceId = faceId,
            FaceMatchThreshold = 70F,
            MaxFaces = 2,
        };

        SearchFacesResponse searchFacesResponse = await
rekognitionClient.SearchFacesAsync(searchFacesRequest);

        Console.WriteLine("Face matching faceId " + faceId);

        Console.WriteLine("Matche(s): ");
        searchFacesResponse.FaceMatches.ForEach(face =>
```

```
        {
            Console.WriteLine($"FaceId: {face.Face.FaceId} Similarity:
{face.Similarity}");
        });
    }
}
```

- For API details, see [SearchFaces](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void searchFaceInCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(new File(sourceImage));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
            .image(souImage)
            .maxFaces(10)
            .faceMatchThreshold(70F)
            .collectionId(collectionId)
            .build();

        SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest) ;
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face: faceImageMatches) {
            System.out.println("The similarity level is " + face.similarity());
            System.out.println();
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [SearchFaces](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollection:  
    """  
        Encapsulates an Amazon Rekognition collection. This class is a thin wrapper  
        around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, collection, rekognition_client):  
        """  
            Initializes a collection object.  
  
            :param collection: Collection data in the format returned by a call to  
                create_collection.  
            :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.collection_id = collection['CollectionId']  
        self.collection_arn, self.face_count, self.created =  
        self._unpack_collection(  
            collection)  
        self.rekognition_client = rekognition_client  
  
    @staticmethod  
    def _unpack_collection(collection):  
        """  
            Unpacks optional parts of a collection that can be returned by  
            describe_collection.  
  
            :param collection: The collection data.  
            :return: A tuple of the data in the collection.  
        """  
        return (  
            collection.get('CollectionArn'),  
            collection.get('FaceCount', 0),  
            collection.get('CreationTimestamp'))  
  
    def search_faces(self, face_id, threshold, max_faces):  
        """  
            Searches for faces in the collection that match another face from the  
            collection.  
  
            :param face_id: The ID of the face in the collection to search for.  
            :param threshold: The match confidence must be greater than this value  
                for a face to be included in the results.  
            :param max_faces: The maximum number of faces to return.  
            :return: The list of matching faces found in the collection. This list does  
                not contain the face specified by `face_id`.  
        """  
        try:  
            response = self.rekognition_client.search_faces(  
                CollectionId=self.collection_id, FaceId=face_id,  
                FaceMatchThreshold=threshold, MaxFaces=max_faces)  
            faces = [RekognitionFace(face['Face']) for face in  
            response['FaceMatches']]  
            logger.info(  
                "Found %s faces in %s that match %s.", len(faces),  
                self.collection_id,  
                face_id)  
        except ClientError:  
            logger.exception(  
                "Couldn't search for faces in %s that match %s.",  
                self.collection_id,  
                face_id)  
            raise  
        else:  
            return faces
```

- For API details, see [SearchFaces in AWS SDK for Python \(Boto3\) API Reference](#).

## Search for faces in an Amazon Rekognition collection compared to a reference image using an AWS SDK

The following code examples show how to search for faces in an Amazon Rekognition collection compared to a reference image.

For more information, see [Searching for a face \(image\)](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

///<summary>
/// Uses the Amazon Rekognition Service to search for images matching those
/// in a collection. The example was created using the AWS SDK for .NET
/// version 3.7 and .NET Core 5.0.
///</summary>
public class SearchFacesMatchingImage
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        string bucket = "bucket";
        string photo = "input.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        // Get an image object from S3 bucket.
        var image = new Image()
        {
            S3Object = new S3Object()
            {
                Bucket = bucket,
                Name = photo,
            },
        };

        var searchFacesByImageRequest = new SearchFacesByImageRequest()
        {
            CollectionId = collectionId,
            Image = image,
            FaceMatchThreshold = 70F,
            MaxFaces = 2,
        };

        SearchFacesByImageResponse searchFacesByImageResponse = await
rekognitionClient.SearchFacesByImageAsync(searchFacesByImageRequest);
```

```
        Console.WriteLine("Faces matching largest face in image from " +  
    photo);  
    searchFacesByImageResponse.FaceMatches.ForEach(face =>  
    {  
        Console.WriteLine($"FaceId: {face.Face.FaceId}, Similarity:  
{face.Similarity}");  
    });  
}
```

- For API details, see [SearchFacesByImage](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void searchFacebyId(RekognitionClient rekClient, String  
collectionId, String faceId) {  
  
    try {  
        SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()  
            .collectionId(collectionId)  
            .faceId(faceId)  
            .faceMatchThreshold(70F)  
            .maxFaces(2)  
            .build();  
  
        SearchFacesResponse imageResponse =  
rekClient.searchFaces(searchFacesRequest) ;  
        System.out.println("Faces matching in the collection");  
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();  
        for (FaceMatch face: faceImageMatches) {  
            System.out.println("The similarity level is " + face.similarity());  
            System.out.println();  
        }  
  
    } catch (RekognitionException e) {  
        System.out.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [SearchFacesByImage](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollection:
```

```

"""
Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
around parts of the Boto3 Amazon Rekognition API.
"""
def __init__(self, collection, rekognition_client):
    """
    Initializes a collection object.

    :param collection: Collection data in the format returned by a call to
                       create_collection.
    :param rekognition_client: A Boto3 Rekognition client.
    """
    self.collection_id = collection['CollectionId']
    self.collection_arn, self.face_count, self.created =
    self._unpack_collection(
        collection)
    self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get('CollectionArn'),
            collection.get('FaceCount', 0),
            collection.get('CreationTimestamp'))

    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.

        :param image: The image that contains the reference face to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: A tuple. The first element is the face found in the reference
                image.
                The second element is the list of matching faces found in the
                collection.
        """
        try:
            response = self.rekognition_client.search_faces_by_image(
                CollectionId=self.collection_id, Image=image.image,
                FaceMatchThreshold=threshold, MaxFaces=max_faces)
            image_face = RekognitionFace({
                'BoundingBox': response['SearchedFaceBoundingBox'],
                'Confidence': response['SearchedFaceConfidence']
            })
            collection_faces = [
                RekognitionFace(face['Face']) for face in response['FaceMatches']]
            logger.info("Found %s faces in the collection that match the largest "
                        "face in %s.", len(collection_faces), image.image_name)
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                image.image_name)
            raise
        else:
            return image_face, collection_faces

```

- For API details, see [SearchFacesByImage](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios for Amazon Rekognition using AWS SDKs

The following code examples show how to use Amazon Rekognition with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Build an Amazon Rekognition collection and find faces in it using an AWS SDK \(p. 1572\)](#)
- [Detect and display elements in images with Amazon Rekognition using an AWS SDK \(p. 1579\)](#)
- [Detect information in videos using Amazon Rekognition and the AWS SDK \(p. 1588\)](#)

### Build an Amazon Rekognition collection and find faces in it using an AWS SDK

The following code example shows how to:

- Create an Amazon Rekognition collection.
- Add images to the collection and detect faces in it.
- Search the collection for faces that match a reference image.
- Delete a collection.

For more information, see [Searching faces in a collection](#).

#### Python

##### SDK for Python (Boto3)

###### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create classes that wrap Amazon Rekognition functions.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
from rekognition_objects import RekognitionFace
from rekognition_image_detection import RekognitionImage

logger = logging.getLogger(__name__)

class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.
    
```

```
:param image: Data that defines the image, either the image bytes or
    an Amazon S3 bucket and object key.
:param image_name: The name of the image.
:param rekognition_client: A Boto3 Rekognition client.
"""
    self.image = image
    self.image_name = image_name
    self.rekognition_client = rekognition_client

@classmethod
def from_file(cls, image_file_name, rekognition_client, image_name=None):
"""
Creates a RekognitionImage object from a local file.

:param image_file_name: The file name of the image. The file is opened and
its
    bytes are read.
:param rekognition_client: A Boto3 Rekognition client.
:param image_name: The name of the image. If this is not specified, the
    file name is used as the image name.
:return: The RekognitionImage object, initialized with image bytes from the
file.
"""
with open(image_file_name, 'rb') as img_file:
    image = {'Bytes': img_file.read()}
name = image_file_name if image_name is None else image_name
return cls(image, name, rekognition_client)

class RekognitionCollectionManager:
"""
Encapsulates Amazon Rekognition collection management functions.
This class is a thin wrapper around parts of the Boto3 Amazon Rekognition API.
"""
def __init__(self, rekognition_client):
"""
Initializes the collection manager object.

:param rekognition_client: A Boto3 Rekognition client.
"""
    self.rekognition_client = rekognition_client

def create_collection(self, collection_id):
"""
Creates an empty collection.

:param collection_id: Text that identifies the collection.
:return: The newly created collection.
"""
try:
    response = self.rekognition_client.create_collection(
        CollectionId=collection_id)
    response['CollectionId'] = collection_id
    collection = RekognitionCollection(response, self.rekognition_client)
    logger.info("Created collection %s.", collection_id)
except ClientError:
    logger.exception("Couldn't create collection %s.", collection_id)
    raise
else:
    return collection

def list_collections(self, max_results):
"""
Lists collections for the current account.

:param max_results: The maximum number of collections to return.
:return: The list of collections for the current account.

```

```
"""
    try:
        response =
self.rekognition_client.list_collections(MaxResults=max_results)
        collections = [
            RekognitionCollection({'CollectionId': col_id},
self.rekognition_client)
            for col_id in response['CollectionIds']]
    except ClientError:
        logger.exception("Couldn't list collections.")
        raise
else:
    return collections

class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection['CollectionId']
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection)
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get('CollectionArn'),
            collection.get('FaceCount', 0),
            collection.get('CreationTimestamp'))

    def to_dict(self):
        """
        Renders parts of the collection data to a dict.

        :return: The collection data as a dict.
        """
        rendering = {
            'collection_id': self.collection_id,
            'collection_arn': self.collection_arn,
            'face_count': self.face_count,
            'created': self.created
        }
        return rendering

    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        
```

```

"""
try:
    response = self.rekognition_client.describe_collection(
        CollectionId=self.collection_id)
    # Work around capitalization of Arn vs. ARN
    response['CollectionArn'] = response.get('CollectionARN')
    (self.collection_arn, self.face_count,
     self.created) = self._unpack_collection(response)
    logger.info("Got data for collection %s.", self.collection_id)
except ClientError:
    logger.exception("Couldn't get data for collection %s.",
                     self.collection_id)
    raise
else:
    return self.to_dict()

def delete_collection(self):
    """
    Deletes the collection.
    """
    try:

        self.rekognition_client.delete_collection(CollectionId=self.collection_id)
        logger.info("Deleted collection %s.", self.collection_id)
        self.collection_id = None
    except ClientError:
        logger.exception("Couldn't delete collection %s.", self.collection_id)
        raise

def index_faces(self, image, max_faces):
    """
    Finds faces in the specified image, indexes them, and stores them in the
    collection.

    :param image: The image to index.
    :param max_faces: The maximum number of faces to index.
    :return: A tuple. The first element is a list of indexed faces.
            The second element is a list of faces that couldn't be indexed.
    """
    try:
        response = self.rekognition_client.index_faces(
            CollectionId=self.collection_id, Image=image.image,
            ExternalImageId=image.image_name, MaxFaces=max_faces,
            DetectionAttributes=['ALL'])
        indexed_faces = [
            RekognitionFace({**face['Face'], **face['FaceDetail']})
            for face in response['FaceRecords']]
        unindexed_faces = [
            RekognitionFace(face['FaceDetail'])
            for face in response['UnindexedFaces']]
        logger.info(
            "Indexed %s faces in %s. Could not index %s faces.",
            len(indexed_faces),
            image.image_name, len(unindexed_faces))
    except ClientError:
        logger.exception("Couldn't index faces in image %s.", image.image_name)
        raise
    else:
        return indexed_faces, unindexed_faces

def list_faces(self, max_results):
    """
    Lists the faces currently indexed in the collection.

    :param max_results: The maximum number of faces to return.
    :return: The list of faces in the collection.
    """

```

```

"""
try:
    response = self.rekognition_client.list_faces(
        CollectionId=self.collection_id, MaxResults=max_results)
    faces = [RekognitionFace(face) for face in response['Faces']]
    logger.info(
        "Found %s faces in collection %s.", len(faces), self.collection_id)
except ClientError:
    logger.exception(
        "Couldn't list faces in collection %s.", self.collection_id)
    raise
else:
    return faces

def search_faces(self, face_id, threshold, max_faces):
    """
    Searches for faces in the collection that match another face from the
    collection.

    :param face_id: The ID of the face in the collection to search for.
    :param threshold: The match confidence must be greater than this value
                      for a face to be included in the results.
    :param max_faces: The maximum number of faces to return.
    :return: The list of matching faces found in the collection. This list does
            not contain the face specified by `face_id`.
    """
    try:
        response = self.rekognition_client.search_faces(
            CollectionId=self.collection_id, FaceId=face_id,
            FaceMatchThreshold=threshold, MaxFaces=max_faces)
        faces = [RekognitionFace(face['Face']) for face in
            response['FaceMatches']]
        logger.info(
            "Found %s faces in %s that match %s.", len(faces),
            self.collection_id,
            face_id)
    except ClientError:
        logger.exception(
            "Couldn't search for faces in %s that match %s.",
            self.collection_id,
            face_id)
        raise
    else:
        return faces

def search_faces_by_image(self, image, threshold, max_faces):
    """
    Searches for faces in the collection that match the largest face in the
    reference image.

    :param image: The image that contains the reference face to search for.
    :param threshold: The match confidence must be greater than this value
                      for a face to be included in the results.
    :param max_faces: The maximum number of faces to return.
    :return: A tuple. The first element is the face found in the reference
            image.
                    The second element is the list of matching faces found in the
                    collection.
    """
    try:
        response = self.rekognition_client.search_faces_by_image(
            CollectionId=self.collection_id, Image=image.image,
            FaceMatchThreshold=threshold, MaxFaces=max_faces)
        image_face = RekognitionFace({
            'BoundingBox': response['SearchedFaceBoundingBox'],
            'Confidence': response['SearchedFaceConfidence']}

```

```

        })
    collection_faces = [
        RekognitionFace(face['Face']) for face in response['FaceMatches']]
    logger.info("Found %s faces in the collection that match the largest "
                "face in %s.", len(collection_faces), image.image_name)
except ClientError:
    logger.exception(
        "Couldn't search for faces in %s that match %s.",
    self.collection_id,
        image.image_name)
    raise
else:
    return image_face, collection_faces

class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""
    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
                     functions.
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
        """
        self.bounding_box = face.get('BoundingBox')
        self.confidence = face.get('Confidence')
        self.landmarks = face.get('Landmarks')
        self.pose = face.get('Pose')
        self.quality = face.get('Quality')
        age_range = face.get('AgeRange')
        if age_range is not None:
            self.age_range = (age_range.get('Low'), age_range.get('High'))
        else:
            self.age_range = None
        self.smile = face.get('Smile', {}).get('Value')
        self.eyeglasses = face.get('Eyeglasses', {}).get('Value')
        self.sunglasses = face.get('Sunglasses', {}).get('Value')
        self.gender = face.get('Gender', {}).get('Value', None)
        self.beard = face.get('Beard', {}).get('Value')
        self.mustache = face.get('Mustache', {}).get('Value')
        self.eyes_open = face.get('EyesOpen', {}).get('Value')
        self.mouth_open = face.get('MouthOpen', {}).get('Value')
        self.emotions = [emo.get('Type') for emo in face.get('Emotions', [])
                        if emo.get('Confidence', 0) > 50]
        self.face_id = face.get('FaceId')
        self.image_id = face.get('ImageId')
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
        rendering = {}
        if self.bounding_box is not None:
            rendering['bounding_box'] = self.bounding_box
        if self.age_range is not None:
            rendering['age'] = f'{self.age_range[0]} - {self.age_range[1]}'
        if self.gender is not None:
            rendering['gender'] = self.gender
        if self.emotions:
            rendering['emotions'] = self.emotions
        if self.face_id is not None:
            rendering['face_id'] = self.face_id

```

```
if self.image_id is not None:
    rendering['image_id'] = self.image_id
if self.timestamp is not None:
    rendering['timestamp'] = self.timestamp
has = []
if self.smile:
    has.append('smile')
if self.eyeglasses:
    has.append('eyeglasses')
if self.sunglasses:
    has.append('sunglasses')
if self.beard:
    has.append('beard')
if self.mustache:
    has.append('mustache')
if self.eyes_open:
    has.append('open eyes')
if self.mouth_open:
    has.append('open mouth')
if has:
    rendering['has'] = has
return rendering
```

Use the wrapper classes to build a collection of faces from a set of images and then search for faces in the collection.

```
def usage_demo():
    print('*'*88)
    print("Welcome to the Amazon Rekognition face collection demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    rekognition_client = boto3.client('rekognition')
    images = [
        RekognitionImage.from_file(
            '.media/pexels-agung-pandit-wiguna-1128316.jpg', rekognition_client,
            image_name='sitting'),
        RekognitionImage.from_file(
            '.media/pexels-agung-pandit-wiguna-1128317.jpg', rekognition_client,
            image_name='hopping'),
        RekognitionImage.from_file(
            '.media/pexels-agung-pandit-wiguna-1128318.jpg', rekognition_client,
            image_name='biking')]

    collection_mgr = RekognitionCollectionManager(rekognition_client)
    collection = collection_mgr.create_collection('doc-example-collection-demo')
    print(f"Created collection {collection.collection_id}:")
    pprint(collection.describe_collection())

    print("Indexing faces from three images:")
    for image in images:
        collection.index_faces(image, 10)
    print("Listing faces in collection:")
    faces = collection.list_faces(10)
    for face in faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    print(f"Searching for faces in the collection that match the first face in the
"
          f"list (Face ID: {faces[0].face_id}.")
    found_faces = collection.search_faces(faces[0].face_id, 80, 10)
```

```
print(f"Found {len(found_faces)} matching faces.")
for face in found_faces:
    pprint(face.to_dict())
input("Press Enter to continue.")

print(f"Searching for faces in the collection that match the largest face in "
      f"{images[0].image_name}.")
image_face, match_faces = collection.search_faces_by_image(images[0], 80, 10)
print(f"The largest face in {images[0].image_name} is:")
pprint(image_face.to_dict())
print(f"Found {len(match_faces)} matching faces.")
for face in match_faces:
    pprint(face.to_dict())
input("Press Enter to continue.")

collection.delete_collection()
print('Thanks for watching!')
print('*'*88)
```

## Detect and display elements in images with Amazon Rekognition using an AWS SDK

The following code example shows how to:

- Detect elements in images by using Amazon Rekognition.
- Display images and draw bounding boxes around detected elements.

For more information, see [Displaying bounding boxes](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create classes to wrap Amazon Rekognition functions.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests

from rekognition_objects import (
    RekognitionFace, RekognitionCelebrity, RekognitionLabel,
    RekognitionModerationLabel, RekognitionText, show_bounding_boxes,
    show_polygons)

logger = logging.getLogger(__name__)

class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, image, image_name, rekognition_client):
```

```
"""
Initializes the image object.

:param image: Data that defines the image, either the image bytes or
    an Amazon S3 bucket and object key.
:param image_name: The name of the image.
:param rekognition_client: A Boto3 Rekognition client.
"""

self.image = image
self.image_name = image_name
self.rekognition_client = rekognition_client

@classmethod
def from_file(cls, image_file_name, rekognition_client, image_name=None):
    """
    Creates a RekognitionImage object from a local file.

    :param image_file_name: The file name of the image. The file is opened and
    its
        bytes are read.
    :param rekognition_client: A Boto3 Rekognition client.
    :param image_name: The name of the image. If this is not specified, the
        file name is used as the image name.
    :return: The RekognitionImage object, initialized with image bytes from the
        file.
    """

    with open(image_file_name, 'rb') as img_file:
        image = {'Bytes': img_file.read()}
        name = image_file_name if image_name is None else image_name
    return cls(image, name, rekognition_client)

@classmethod
def from_bucket(cls, s3_object, rekognition_client):
    """
    Creates a RekognitionImage object from an Amazon S3 object.

    :param s3_object: An Amazon S3 object that identifies the image. The image
        is not retrieved until needed for a later call.
    :param rekognition_client: A Boto3 Rekognition client.
    :return: The RekognitionImage object, initialized with Amazon S3 object
    data.
    """

    image = {'S3Object': {'Bucket': s3_object.bucket_name, 'Name':
    s3_object.key}}
    return cls(image, s3_object.key, rekognition_client)

def detect_faces(self):
    """
    Detects faces in the image.

    :return: The list of faces found in the image.
    """

    try:
        response = self.rekognition_client.detect_faces(
            Image=self.image, Attributes=['ALL'])
        faces = [RekognitionFace(face) for face in response['FaceDetails']]
        logger.info("Detected %s faces.", len(faces))
    except ClientError:
        logger.exception("Couldn't detect faces in %s.", self.image_name)
        raise
    else:
        return faces

def detect_labels(self, max_labels):
    """
    Detects labels in the image. Labels are objects and people.
```

```
:param max_labels: The maximum number of labels to return.
:return: The list of labels detected in the image.
"""
try:
    response = self.rekognition_client.detect_labels(
        Image=self.image, MaxLabels=max_labels)
    labels = [RekognitionLabel(label) for label in response['Labels']]
    logger.info("Found %s labels in %s.", len(labels), self.image_name)
except ClientError:
    logger.info("Couldn't detect labels in %s.", self.image_name)
    raise
else:
    return labels

def recognize_celebrities(self):
    """
    Detects celebrities in the image.

    :return: A tuple. The first element is the list of celebrities found in
            the image. The second element is the list of faces that were
            detected but did not match any known celebrities.
    """
    try:
        response = self.rekognition_client.recognize_celebrities(
            Image=self.image)
        celebrities = [RekognitionCelebrity(celeb)
                       for celeb in response['CelebrityFaces']]
        other_faces = [RekognitionFace(face)
                       for face in response['UnrecognizedFaces']]
        logger.info(
            "Found %s celebrities and %s other faces in %s.", len(celebrities),
            len(other_faces), self.image_name)
    except ClientError:
        logger.exception("Couldn't detect celebrities in %s.", self.image_name)
        raise
    else:
        return celebrities, other_faces

def compare_faces(self, target_image, similarity):
    """
    Compares faces in the image with the largest face in the target image.

    :param target_image: The target image to compare against.
    :param similarity: Faces in the image must have a similarity value greater
                       than this value to be included in the results.
    :return: A tuple. The first element is the list of faces that match the
            reference image. The second element is the list of faces that have
            a similarity value below the specified threshold.
    """
    try:
        response = self.rekognition_client.compare_faces(
            SourceImage=self.image,
            TargetImage=target_image.image,
            SimilarityThreshold=similarity)
        matches = [RekognitionFace(match['Face']) for match
                   in response['FaceMatches']]
        unmatches = [RekognitionFace(face) for face in
                    response['UnmatchedFaces']]
        logger.info(
            "Found %s matched faces and %s unmatched faces.",
            len(matches), len(unmatches))
    except ClientError:
        logger.exception(
            "Couldn't match faces from %s to %s.", self.image_name,
            target_image.image_name)
```

```

        raise
    else:
        return matches, unmatches

def detect_moderation_labels(self):
    """
    Detects moderation labels in the image. Moderation labels identify content
    that may be inappropriate for some audiences.

    :return: The list of moderation labels found in the image.
    """
    try:
        response = self.rekognition_client.detect_moderation_labels(
            Image=self.image)
        labels = [RekognitionModerationLabel(label)
                  for label in response['ModerationLabels']]
        logger.info(
            "Found %s moderation labels in %s.", len(labels), self.image_name)
    except ClientError:
        logger.exception(
            "Couldn't detect moderation labels in %s.", self.image_name)
        raise
    else:
        return labels

def detect_text(self):
    """
    Detects text in the image.

    :return: The list of text elements found in the image.
    """
    try:
        response = self.rekognition_client.detect_text(Image=self.image)
        texts = [RekognitionText(text) for text in response['TextDetections']]
        logger.info("Found %s texts in %s.", len(texts), self.image_name)
    except ClientError:
        logger.exception("Couldn't detect text in %s.", self.image_name)
        raise
    else:
        return texts

```

Create helper functions to draw bounding boxes and polygons.

```

import io
import logging
from PIL import Image, ImageDraw

logger = logging.getLogger(__name__)

def show_bounding_boxes(image_bytes, box_sets, colors):
    """
    Draws bounding boxes on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param box_sets: A list of lists of bounding boxes to draw on the image.
    :param colors: A list of colors to use to draw the bounding boxes.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for boxes, color in zip(box_sets, colors):
        for box in boxes:
            left = image.width * box['Left']
            top = image.height * box['Top']

```

```

        right = (image.width * box['Width']) + left
        bottom = (image.height * box['Height']) + top
        draw.rectangle([left, top, right, bottom], outline=color, width=3)
    image.show()

def show_polygons(image_bytes, polygons, color):
    """
    Draws polygons on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param polygons: The list of polygons to draw on the image.
    :param color: The color to use to draw the polygons.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for polygon in polygons:
        draw.polygon([
            (image.width * point['X'], image.height * point['Y']) for point in
            polygon],
            outline=color)
    image.show()

```

Create classes to parse objects returned by Amazon Rekognition.

```

class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""
    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
                    functions.
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
        """
        self.bounding_box = face.get('BoundingBox')
        self.confidence = face.get('Confidence')
        self.landmarks = face.get('Landmarks')
        self.pose = face.get('Pose')
        self.quality = face.get('Quality')
        age_range = face.get('AgeRange')
        if age_range is not None:
            self.age_range = (age_range.get('Low'), age_range.get('High'))
        else:
            self.age_range = None
        self.smile = face.get('Smile', {}).get('Value')
        self.eyeglasses = face.get('Eyeglasses', {}).get('Value')
        self.sunglasses = face.get('Sunglasses', {}).get('Value')
        self.gender = face.get('Gender', {}).get('Value', None)
        self.beard = face.get('Beard', {}).get('Value')
        self.mustache = face.get('Mustache', {}).get('Value')
        self.eyes_open = face.get('EyesOpen', {}).get('Value')
        self.mouth_open = face.get('MouthOpen', {}).get('Value')
        self.emotions = [emo.get('Type') for emo in face.get('Emotions', [])
                        if emo.get('Confidence', 0) > 50]
        self.face_id = face.get('FaceId')
        self.image_id = face.get('ImageId')
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """

```

```
"""
rendering = {}
if self.bounding_box is not None:
    rendering['bounding_box'] = self.bounding_box
if self.age_range is not None:
    rendering['age'] = f'{self.age_range[0]} - {self.age_range[1]}'
if self.gender is not None:
    rendering['gender'] = self.gender
if self.emotions:
    rendering['emotions'] = self.emotions
if self.face_id is not None:
    rendering['face_id'] = self.face_id
if self.image_id is not None:
    rendering['image_id'] = self.image_id
if self.timestamp is not None:
    rendering['timestamp'] = self.timestamp
has = []
if self.smile:
    has.append('smile')
if self.eyeglasses:
    has.append('eyeglasses')
if self.sunglasses:
    has.append('sunglasses')
if self.beard:
    has.append('beard')
if self.mustache:
    has.append('mustache')
if self.eyes_open:
    has.append('open eyes')
if self.mouth_open:
    has.append('open mouth')
if has:
    rendering['has'] = has
return rendering

class RekognitionCelebrity:
    """Encapsulates an Amazon Rekognition celebrity."""
    def __init__(self, celebrity, timestamp=None):
        """
        Initializes the celebrity object.

        :param celebrity: Celebrity data, in the format returned by Amazon
        Rekognition
        functions.
        :param timestamp: The time when the celebrity was detected, if the
        celebrity
                    was detected in a video.
        """
        self.info_urls = celebrity.get('Urls')
        self.name = celebrity.get('Name')
        self.id = celebrity.get('Id')
        self.face = RekognitionFace(celebrity.get('Face'))
        self.confidence = celebrity.get('MatchConfidence')
        self.bounding_box = celebrity.get('BoundingBox')
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the celebrity data to a dict.

        :return: A dict that contains the celebrity data.
        """
        rendering = self.face.to_dict()
        if self.name is not None:
            rendering['name'] = self.name
        if self.info_urls:
```

```
        rendering['info URLs'] = self.info_urls
    if self.timestamp is not None:
        rendering['timestamp'] = self.timestamp
    return rendering

class RekognitionPerson:
    """Encapsulates an Amazon Rekognition person."""
    def __init__(self, person, timestamp=None):
        """
        Initializes the person object.

        :param person: Person data, in the format returned by Amazon Rekognition
                       functions.
        :param timestamp: The time when the person was detected, if the person
                          was detected in a video.
        """
        self.index = person.get('Index')
        self.bounding_box = person.get('BoundingBox')
        face = person.get('Face')
        self.face = RekognitionFace(face) if face is not None else None
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the person data to a dict.

        :return: A dict that contains the person data.
        """
        rendering = self.face.to_dict() if self.face is not None else {}
        if self.index is not None:
            rendering['index'] = self.index
        if self.bounding_box is not None:
            rendering['bounding_box'] = self.bounding_box
        if self.timestamp is not None:
            rendering['timestamp'] = self.timestamp
        return rendering

class RekognitionLabel:
    """Encapsulates an Amazon Rekognition label."""
    def __init__(self, label, timestamp=None):
        """
        Initializes the label object.

        :param label: Label data, in the format returned by Amazon Rekognition
                      functions.
        :param timestamp: The time when the label was detected, if the label
                          was detected in a video.
        """
        self.name = label.get('Name')
        self.confidence = label.get('Confidence')
        self.instances = label.get('Instances')
        self.parents = label.get('Parents')
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the label data to a dict.

        :return: A dict that contains the label data.
        """
        rendering = {}
        if self.name is not None:
            rendering['name'] = self.name
        if self.timestamp is not None:
            rendering['timestamp'] = self.timestamp
        return rendering
```

```
class RekognitionModerationLabel:  
    """Encapsulates an Amazon Rekognition moderation label."""  
    def __init__(self, label, timestamp=None):  
        """  
        Initializes the moderation label object.  
  
        :param label: Label data, in the format returned by Amazon Rekognition  
                     functions.  
        :param timestamp: The time when the moderation label was detected, if the  
                          label was detected in a video.  
        """  
        self.name = label.get('Name')  
        self.confidence = label.get('Confidence')  
        self.parent_name = label.get('ParentName')  
        self.timestamp = timestamp  
  
    def to_dict(self):  
        """  
        Renders some of the moderation label data to a dict.  
  
        :return: A dict that contains the moderation label data.  
        """  
        rendering = {}  
        if self.name is not None:  
            rendering['name'] = self.name  
        if self.parent_name is not None:  
            rendering['parent_name'] = self.parent_name  
        if self.timestamp is not None:  
            rendering['timestamp'] = self.timestamp  
        return rendering  
  
class RekognitionText:  
    """Encapsulates an Amazon Rekognition text element."""  
    def __init__(self, text_data):  
        """  
        Initializes the text object.  
  
        :param text_data: Text data, in the format returned by Amazon Rekognition  
                         functions.  
        """  
        self.text = text_data.get('DetectedText')  
        self.kind = text_data.get('Type')  
        self.id = text_data.get('Id')  
        self.parent_id = text_data.get('ParentId')  
        self.confidence = text_data.get('Confidence')  
        self.geometry = text_data.get('Geometry')  
  
    def to_dict(self):  
        """  
        Renders some of the text data to a dict.  
  
        :return: A dict that contains the text data.  
        """  
        rendering = {}  
        if self.text is not None:  
            rendering['text'] = self.text  
        if self.kind is not None:  
            rendering['kind'] = self.kind  
        if self.geometry is not None:  
            rendering['polygon'] = self.geometry.get('Polygon')  
        return rendering
```

Use the wrapper classes to detect elements in images and display their bounding boxes. The images used in this example can be found on GitHub along with instructions and more code.

```
def usage_demo():
    print('*'*88)
    print("Welcome to the Amazon Rekognition image detection demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    rekognition_client = boto3.client('rekognition')
    street_scene_file_name = ".media/pexels-kaique-rocha-109919.jpg"
    celebrity_file_name = ".media/pexels-pixabay-53370.jpg"
    one_girl_url = 'https://dhei5unw3vrsx.cloudfront.net/images/source3_resized.jpg'
    three_girls_url = 'https://dhei5unw3vrsx.cloudfront.net/images/target3_resized.jpg'
    swimwear_object = boto3.resource('s3').Object(
        'console-sample-images-pdx', 'yoga_swimwear.jpg')
    book_file_name = '.media/pexels-christina-morillo-1181671.jpg'

    street_scene_image = RekognitionImage.from_file(
        street_scene_file_name, rekognition_client)
    print(f"Detecting faces in {street_scene_image.image_name}...")
    faces = street_scene_image.detect_faces()
    print(f"Found {len(faces)} faces, here are the first three.")
    for face in faces[:3]:
        pprint(face.to_dict())
    show_bounding_boxes(
        street_scene_image.image['Bytes'], [[face.bounding_box for face in faces]],
        ['aqua'])
    input("Press Enter to continue.")

    print(f"Detecting labels in {street_scene_image.image_name}...")
    labels = street_scene_image.detect_labels(100)
    print(f"Found {len(labels)} labels.")
    for label in labels:
        pprint(label.to_dict())
    names = []
    box_sets = []
    colors = ['aqua', 'red', 'white', 'blue', 'yellow', 'green']
    for label in labels:
        if label.instances:
            names.append(label.name)
            box_sets.append([inst['BoundingBox'] for inst in label.instances])
    print(f"Showing bounding boxes for {names} in {colors[:len(names)]}.")
    show_bounding_boxes(
        street_scene_image.image['Bytes'], box_sets, colors[:len(names)])
    input("Press Enter to continue.")

    celebrity_image = RekognitionImage.from_file(
        celebrity_file_name, rekognition_client)
    print(f"Detecting celebrities in {celebrity_image.image_name}...")
    celebs, others = celebrity_image.recognize_celebrities()
    print(f"Found {len(celebs)} celebrities.")
    for celeb in celebs:
        pprint(celeb.to_dict())
    show_bounding_boxes(
        celebrity_image.image['Bytes'],
        [[celeb.face.bounding_box for celeb in celebs]], ['aqua'])
    input("Press Enter to continue.")

    girl_image_response = requests.get(one_girl_url)
    girl_image = RekognitionImage(
        {'Bytes': girl_image_response.content}, "one-girl", rekognition_client)
    group_image_response = requests.get(three_girls_url)
```

```
group_image = RekognitionImage(  
    {'Bytes': group_image_response.content}, "three-girls", rekognition_client)  
print("Comparing reference face to group of faces...")  
matches, unmatches = girl_image.compare_faces(group_image, 80)  
print(f"Found {len(matches)} face matching the reference face.")  
show_bounding_boxes(  
    group_image.image['Bytes'], [[match.bounding_box for match in matches]],  
    ['aqua'])  
input("Press Enter to continue.")  
  
swimwear_image = RekognitionImage.from_bucket(swimwear_object,  
rekognition_client)  
print(f"Detecting suggestive content in {swimwear_object.key}...")  
labels = swimwear_image.detect_moderation_labels()  
print(f"Found {len(labels)} moderation labels.")  
for label in labels:  
    pprint(label.to_dict())  
input("Press Enter to continue.")  
  
book_image = RekognitionImage.from_file(book_file_name, rekognition_client)  
print(f"Detecting text in {book_image.image_name}...")  
texts = book_image.detect_text()  
print(f"Found {len(texts)} text instances. Here are the first seven:")  
for text in texts[:7]:  
    pprint(text.to_dict())  
show_polygons(  
    book_image.image['Bytes'], [text.geometry['Polygon'] for text in texts],  
    'aqua')  
  
print("Thanks for watching!")  
print('-'*88)
```

## Detect information in videos using Amazon Rekognition and the AWS SDK

The following code examples show how to:

- Start Amazon Rekognition jobs to detect elements like people, objects, and text in videos.
- Check job status until jobs finish.
- Output the list of elements detected by each job.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get celebrity results from a video located in an Amazon S3 bucket.

```
public static void StartCelebrityDetection(RekognitionClient rekClient,  
                                         NotificationChannel channel,  
                                         String bucket,  
                                         String video){  
    try {  
        S3Object s3obj = S3Object.builder()  
            .bucket(bucket)  
            .name(video)
```

```
.build();

Video vid0b = Video.builder()
    .s3Object(s3obj)
    .build();

StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
    .jobTag("Celebrities")
    .notificationChannel(channel)
    .video(vid0b)
    .build();

StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient.startCelebrityRecognition(recognitionRequest);
startJobId = startCelebrityRecognitionResult.jobId();

} catch(RekognitionException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}

public static void GetCelebrityDetectionResults(RekognitionClient rekClient) {

try {
    String paginationToken=null;
    GetCelebrityRecognitionResponse recognitionResponse = null;
    boolean finished = false;
    String status;
    int yy=0 ;

    do{
        if (recognitionResponse !=null)
            paginationToken = recognitionResponse.nextToken();

        GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
    .jobId(startJobId)
    .nextToken(paginationToken)
    .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
    .maxResults(10)
    .build();

        // Wait until the job succeeds
        while (!finished) {
            recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
            status = recognitionResponse.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }
        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null.
        VideoMetadata videoMetaData=recognitionResponse.videoMetadata();
        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
    }
}
```

```
System.out.println("FrameRate: " + videoMetaData.frameRate());
System.out.println("Job");

List<CelebrityRecognition> celebs=
recognitionResponse.celebrities();
for (CelebrityRecognition celeb: celebs) {
    long seconds=celeb.timestamp()/1000;
    System.out.print("Sec: " + seconds + " ");
    CelebrityDetail details=celeb.celebrity();
    System.out.println("Name: " + details.name());
    System.out.println("Id: " + details.id());
    System.out.println();
}
} while (recognitionResponse.nextToken() != null);

} catch(RekognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

Detect labels in a video by a label detection operation.

```
public static void startLabels(RekognitionClient rekClient,
                               NotificationChannel channel,
                               String bucket,
                               String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vid0b = Video.builder()
            .s3Object(s3obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vid0b)
            .minConfidence(50F)
            .build();

        StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

        boolean ans = true;
        String status = "";
        int yy = 0;
        while (ans) {

            GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                . jobId(startJobId)
                . maxResults(10)
                . build();

            GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
            status = result.jobStatusAsString();
    }
}
```

```
        if (status.compareTo("SUCCEEDED") == 0)
            ans = false;
        else
            System.out.println(yy +" status is: "+status);

            Thread.sleep(1000);
            yy++;
    }

    System.out.println(startJobId +" status is: "+status);

} catch(RekognitionException | InterruptedException e) {
    e.getMessage();
    System.exit(1);
}
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {

    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message: messages) {
                String notification = message.body();

                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonMessageTree = mapper.readTree(notification);
                JsonNode messageBodyText = jsonMessageTree.get("Message");
                ObjectMapper operationResultMapper = new ObjectMapper();
                JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                JsonNode operationJobId = jsonResultTree.get("JobId");
                JsonNode operationStatus = jsonResultTree.get("Status");
                System.out.println("Job found in JSON is " + operationJobId);

                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

                String jobId = operationJobId.textValue();
                if (startJobId.compareTo(jobId)==0) {
                    System.out.println("Job id: " + operationJobId );
                    System.out.println("Status : " +
operationStatus.toString());

                    if (operationStatus.asText().equals("SUCCEEDED"))
                        GetResultsLabels(rekClient);
                    else
                        System.out.println("Video analysis failed");

                    sqs.deleteMessage(deleteMessageRequest);
                }
                else{
                    System.out.println("Job received was not job " +
startJobId);
                }
            }
        }
    }
}
```

```
        sqs.deleteMessage(deleteMessageRequest);
    }
}

} catch(RekognitionException e) {
    e.getMessage();
    System.exit(1);
} catch (JsonMappingException e) {
    e.printStackTrace();
} catch (JsonProcessingException e) {
    e.printStackTrace();
}
}

// Gets the job results by calling GetLabelDetection
private static void GetResultsLabels(RekognitionClient rekClient) {

    int maxResults=10;
    String paginationToken=null;
    GetLabelDetectionResponse labelDetectionResult=null;

    try {
        do {
            if (labelDetectionResult !=null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest=
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
                .maxResults(maxResults)
                .nextToken(paginationToken)
                .build();

            labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
            VideoMetadata videoMetaData=labelDetectionResult.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());

            List<LabelDetection> detectedLabels= labelDetectionResult.labels();
            for (LabelDetection detectedLabel: detectedLabels) {
                long seconds=detectedLabel.timestamp();
                Label label=detectedLabel.label();
                System.out.println("Millisecond: " + seconds + " ");

                System.out.println("    Label:" + label.name());
                System.out.println("    Confidence:" +
detectedLabel.label().confidence().toString());

                List<Instance> instances = label.instances();
                System.out.println("    Instances of " + label.name());

                if (instances.isEmpty()) {
                    System.out.println("        " + "None");
                } else {
                    for (Instance instance : instances) {
                        System.out.println("            Confidence: " +
instance.confidence().toString());
                        System.out.println("            Bounding box: " +
instance.boundingBox().toString());
                    }
                }
            }
        }
    }
}
```

```
        }
        System.out.println("    Parent labels for " + label.name() +
        ":");

        List<Parent> parents = label.parents();

        if (parents.isEmpty()) {
            System.out.println("        None");
        } else {
            for (Parent parent : parents) {
                System.out.println("        " + parent.name());
            }
        }
        System.out.println();
    }

} while (labelDetectionResult !=null &&
labelDetectionResult.nextToken() != null);

}

} catch(RekognitionException e) {
    e.getMessage();
    System.exit(1);
}
```

Detect faces in a video stored in an Amazon S3 bucket.

```
public static void startLabels(RekognitionClient rekClient,
                               NotificationChannel channel,
                               String bucket,
                               String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidOb)
            .minConfidence(50F)
            .build();

        StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

        boolean ans = true;
        String status = "";
        int yy = 0;
        while (ans) {

            GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .maxResults(10)
                .build();

            GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
```

```
        status = result.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            ans = false;
        else
            System.out.println(yy +" status is: "+status);

        Thread.sleep(1000);
        yy++;
    }

    System.out.println(startJobId +" status is: "+status);

} catch(RekognitionException | InterruptedException e) {
    e.getMessage();
    System.exit(1);
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {

    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message: messages) {
                String notification = message.body();

                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonMessageTree = mapper.readTree(notification);
                JsonNode messageBodyText = jsonMessageTree.get("Message");
                ObjectMapper operationResultMapper = new ObjectMapper();
                JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                JsonNode operationJobId = jsonResultTree.get("JobId");
                JsonNode operationStatus = jsonResultTree.get("Status");
                System.out.println("Job found in JSON is " + operationJobId);

                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

                String jobId = operationJobId.textValue();
                if (startJobId.compareTo(jobId)==0) {
                    System.out.println("Job id: " + operationJobId );
                    System.out.println("Status : " +
operationStatus.toString());

                    if (operationStatus.asText().equals("SUCCEEDED"))
                        GetResultsLabels(rekClient);
                    else
                        System.out.println("Video analysis failed");

                    sqs.deleteMessage(deleteMessageRequest);
                }
            }
        }
    }
}
```

```

        System.out.println("Job received was not job " +
startJobId);
        sqs.deleteMessage(deleteMessageRequest);
    }
}

} catch(RekognitionException e) {
    e.getMessage();
    System.exit(1);
} catch (JsonMappingException e) {
    e.printStackTrace();
} catch (JsonProcessingException e) {
    e.printStackTrace();
}
}

// Gets the job results by calling GetLabelDetection
private static void GetResultsLabels(RekognitionClient rekClient) {

    int maxResults=10;
    String paginationToken=null;
    GetLabelDetectionResponse labelDetectionResult=null;

    try {
        do {
            if (labelDetectionResult !=null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest=
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
                .maxResults(maxResults)
                .nextToken(paginationToken)
                .build();

            labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
            VideoMetadata videoMetaData=labelDetectionResult.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());

            List<LabelDetection> detectedLabels= labelDetectionResult.labels();
            for (LabelDetection detectedLabel: detectedLabels) {
                long seconds=detectedLabel.timestamp();
                Label label=detectedLabel.label();
                System.out.println("Millisecond: " + seconds + " ");

                System.out.println("    Label:" + label.name());
                System.out.println("    Confidence:" +
detectedLabel.label().confidence().toString());

                List<Instance> instances = label.instances();
                System.out.println("    Instances of " + label.name());

                if (instances.isEmpty()) {
                    System.out.println("        " + "None");
                } else {
                    for (Instance instance : instances) {
                        System.out.println("        Confidence: " +
instance.confidence().toString());
                }
            }
        }
    }
}

```

```

        System.out.println("      Bounding box: " +
instance.boundingBox().toString());
    }
}
System.out.println("  Parent labels for " + label.name() +
":");
List<Parent> parents = label.parents();

if (parents.isEmpty()) {
    System.out.println("      None");
} else {
    for (Parent parent : parents) {
        System.out.println("      " + parent.name());
    }
}
System.out.println();
}
} while (labelDetectionResult !=null &&
labelDetectionResult.nextToken() != null);

} catch(RekognitionException e) {
    e.getMessage();
    System.exit(1);
}
}
}

```

Detect inappropriate or offensive content in a video stored in an Amazon S3 bucket.

```

public static void startModerationDetection(RekognitionClient rekClient,
                                            NotificationChannel channel,
                                            String bucket,
                                            String video) {

    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
            .jobTag("Moderation")
            .notificationChannel(channel)
            .video(vidOb)
            .build();

        StartContentModerationResponse startModDetectionResult =
rekClient.startContentModeration(modDetectionRequest);
        startJobId=startModDetectionResult.jobId();

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void GetModResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;

```

```

GetContentModerationResponse modDetectionResponse=null;
boolean finished = false;
String status;
int yy=0 ;

do{
    if (modDetectionResponse !=null)
        paginationToken = modDetectionResponse.nextToken();

    GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
    .jobId(startJobId)
    .nextToken(paginationToken)
    .maxResults(10)
    .build();

    // Wait until the job succeeds
    while (!finished) {
        modDetectionResponse =
rekClient.getContentModeration(modRequest);
        status = modDetectionResponse.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null
    VideoMetadata videoMetaData=modDetectionResponse.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " + videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
    for (ContentModerationDetection mod: mods) {
        long seconds=mod.timestamp()/1000;
        System.out.print("Mod label: " + seconds + " ");
        System.out.println(mod.moderationLabel().toString());
        System.out.println();
    }
}

} while (modDetectionResponse !=null &&
modDetectionResponse.nextToken() != null);

} catch(RekognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
}

```

Detect technical cue segments and shot detection segments in a video stored in an Amazon S3 bucket.

```
public static void StartSegmentDetection (RekognitionClient rekClient,
```

```

        NotificationChannel channel,
        String bucket,
        String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartShotDetectionFilter cueDetectionFilter =
StartShotDetectionFilter.builder()
            .minSegmentConfidence(60F)
            .build();

        StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
StartTechnicalCueDetectionFilter.builder()
            .minSegmentConfidence(60F)
            .build();

        StartSegmentDetectionFilters filters =
StartSegmentDetectionFilters.builder()
            .shotFilter(cueDetectionFilter)
            .technicalCueFilter(technicalCueDetectionFilter)
            .build();

        StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .segmentTypes(SegmentType.TECHNICAL_CUE , SegmentType.SHOT)
            .video(vidOb)
            .filters(filters)
            .build();

        StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
        startJobId = segDetectionResponse.jobId();

    } catch(RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getSegmentResults(RekognitionClient rekClient) {
    try {
        String paginationToken = null;
        GetSegmentDetectionResponse segDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (segDetectionResponse != null)
                paginationToken = segDetectionResponse.nextToken();

            GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
            .jobId(startJobId)
            .nextToken(paginationToken)
            .maxResults(10)
    
```

```

        .build();

        // Wait until the job succeeds.
        while (!finished) {
            segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
            status = segDetectionResponse.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }
        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null.
        List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
        for (VideoMetadata metaData : videoMetaData) {
            System.out.println("Format: " + metaData.format());
            System.out.println("Codec: " + metaData.codec());
            System.out.println("Duration: " + metaData.durationMillis());
            System.out.println("FrameRate: " + metaData.frameRate());
            System.out.println("Job");
        }

        List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
        for (SegmentDetection detectedSegment : detectedSegments) {
            String type = detectedSegment.type().toString();
            if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
                System.out.println("Technical Cue");
                TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
                System.out.println("\tType: " + segmentCue.type());
                System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
            }

            if (type.contains(SegmentType.SHOT.toString())) {
                System.out.println("Shot");
                ShotSegment segmentShot = detectedSegment.shotSegment();
                System.out.println("\tIndex " + segmentShot.index());
                System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
            }

            long seconds = detectedSegment.durationMillis();
            System.out.println("\tDuration : " + seconds + "
milliseconds");
            System.out.println("\tStart time code: " +
detectedSegment.startTimecodeSMPTE());
            System.out.println("\tEnd time code: " +
detectedSegment.endTimecodeSMPTE());
            System.out.println("\tDuration time code: " +
detectedSegment.durationSMPTE());
            System.out.println();
        }

    } while (segDetectionResponse !=null &&
segDetectionResponse.nextToken() != null);

} catch(RekognitionException | InterruptedException e) {

```

```
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

Detect text in a video stored in a video stored in an Amazon S3 bucket.

```
public static void startTextLabels(RekognitionClient rekClient,
                                    NotificationChannel channel,
                                    String bucket,
                                    String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidOb)
            .build();

        StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void GetTextResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetTextDetectionResponse textDetectionResponse=null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (textDetectionResponse !=null)
                paginationToken = textDetectionResponse.nextToken();

            GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
            .jobId(startJobId)
            .nextToken(paginationToken)
            .maxResults(10)
            .build();

            // Wait until the job succeeds.
            while (!finished) {
                textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                status = textDetectionResponse.jobStatusAsString();
            }
        }
    }
}
```

```

        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    VideoMetadata videoMetaData=textDetectionResponse.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " + videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<TextDetectionResult> labels=
textDetectionResponse.textDetections();
    for (TextDetectionResult detectedText: labels) {
        System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
        System.out.println("Id : " +
detectedText.textDetection().id());
        System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
        System.out.println("Type: " +
detectedText.textDetection().type());
        System.out.println("Text: " +
detectedText.textDetection().detectedText());
        System.out.println();
    }

} while (textDetectionResponse !=null &&
textDetectionResponse.nextToken() != null);

} catch(RekognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}

```

Detect people in a video stored in a video stored in an Amazon S3 bucket.

```

public static void startPersonLabels(RekognitionClient rekClient,
                                    NotificationChannel channel,
                                    String bucket,
                                    String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
            .jobTag("DetectingLabels")
            .video(vidOb)
    }
}

```

```
.notificationChannel(channel)
.build();

StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
startJobId = labelDetectionResponse.jobId();

} catch(RekognitionException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}

public static void GetPersonDetectionResults(RekognitionClient rekClient) {

try {
    String paginationToken=null;
    GetPersonTrackingResponse personTrackingResult=null;
    boolean finished = false;
    String status;
    int yy=0 ;

    do{
        if (personTrackingResult !=null)
            paginationToken = personTrackingResult.nextToken();

        GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                    .jobId(startJobId)
                    .nextToken(paginationToken)
                    .maxResults(10)
                    .build();

        // Wait until the job succeeds
        while (!finished) {

            personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
            status = personTrackingResult.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }

        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null
        VideoMetadata videoMetaData = personTrackingResult.videoMetadata();

        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());
        System.out.println("Job");

        List<PersonDetection> detectedPersons=
personTrackingResult.persons();
        for (PersonDetection detectedPerson: detectedPersons) {

            long seconds=detectedPerson.timestamp()/1000;
            System.out.print("Sec: " + seconds + " ");
        }
    }
}
```

```
        System.out.println("Person Identifier: " +
detectedPerson.person().index());
        System.out.println();
    }

} while (personTrackingResult !=null &&
personTrackingResult.nextToken() != null);

} catch(RekognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [GetCelebrityRecognition](#)
  - [GetContentModeration](#)
  - [GetLabelDetection](#)
  - [GetPersonTracking](#)
  - [GetSegmentDetection](#)
  - [GetTextDetection](#)
  - [StartCelebrityRecognition](#)
  - [StartContentModeration](#)
  - [StartLabelDetection](#)
  - [StartPersonTracking](#)
  - [StartSegmentDetection](#)
  - [StartTextDetection](#)

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Detect faces in a video stored in an Amazon S3 bucket.

```
suspend fun startFaceDetection(channelVal: NotificationChannel?, bucketVal: String,
videoVal: String) {

    val s3Obj = S3Object {
        bucket = bucketVal
        name = videoVal
    }
    val vidOb = Video {
        s3Object = s3Obj
    }

    val request = StartFaceDetectionRequest {
        jobTag = "Faces"
        faceAttributes = FaceAttributes.All
        notificationChannel = channelVal
        video = vidOb
    }
}
```

```

        }

        RekognitionClient { region = "us-east-1" }.use { rekClient ->
            val startLabelDetectionResult = rekClient.startFaceDetection(request)
            startJobId = startLabelDetectionResult.jobId.toString()
        }
    }

suspend fun getFaceResults() {

    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var response: GetFaceDetectionResponse? = null

        val recognitionRequest = GetFaceDetectionRequest {
            jobId = startJobId
            maxResults = 10
        }

        // Wait until the job succeeds.
        while (!finished) {
            response = rekClient.getFaceDetection(recognitionRequest)
            status = response.jobStatus.toString()
            if (status.compareTo("SUCCEEDED") == 0)
                finished = true
            else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = response?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        // Show face information.
        response?.faces?.forEach { face ->
            println("Age: ${face.face?.ageRange}")
            println("Face: ${face.face?.beard}")
            println("Eye glasses: ${face?.face?.eyeglasses}")
            println("Mustache: ${face.face?.mustache}")
            println("Smile: ${face.face?.smile}")
        }
    }
}

```

Detect inappropriate or offensive content in a video stored in an Amazon S3 bucket.

```

suspend fun startModerationDetection(channel: NotificationChannel?, bucketVal: String?, videoVal: String?) {

    val s3Obj = S3Object {
        bucket = bucketVal
        name = videoVal
    }
    val vidOb = Video {
        s3Object = s3Obj
    }

```

```

    val request = StartContentModerationRequest {
        jobTag = "Moderation"
        notificationChannel = channel
        video = vidOb
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val startModDetectionResult = rekClient.startContentModeration(request)
        startJobId = startModDetectionResult.jobId.toString()
    }
}

suspend fun getModResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var modDetectionResponse: GetContentModerationResponse? = null

        val modRequest = GetContentModerationRequest {
            jobId = startJobId
            maxResults = 10
        }

        // Wait until the job succeeds.
        while (!finished) {
            modDetectionResponse = rekClient.getContentModeration(modRequest)
            status = modDetectionResponse.jobStatus.toString()
            if (status.compareTo("SUCCEEDED") == 0)
                finished = true
            else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = modDetectionResponse?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        modDetectionResponse?.moderationLabels?.forEach { mod ->
            val seconds: Long = mod.timestamp / 1000
            print("Mod label: $seconds ")
            println(mod.moderationLabel)
        }
    }
}

```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - [GetCelebrityRecognition](#)
  - [GetContentModeration](#)
  - [GetLabelDetection](#)
  - [GetPersonTracking](#)
  - [GetSegmentDetection](#)
  - [GetTextDetection](#)
  - [StartCelebrityRecognition](#)
  - [StartContentModeration](#)

- [StartLabelDetection](#)
- [StartPersonTracking](#)
- [StartSegmentDetection](#)
- [StartTextDetection](#)

## Cross-service examples for Amazon Rekognition using AWS SDKs

The following code examples show how to use Amazon Rekognition with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Detect PPE in images with Amazon Rekognition using an AWS SDK \(p. 1606\)](#)
- [Detect faces in an image using an AWS SDK \(p. 1607\)](#)
- [Detect objects in images with Amazon Rekognition using an AWS SDK \(p. 1607\)](#)
- [Detect people and objects in a video with Amazon Rekognition using an AWS SDK \(p. 1610\)](#)
- [Save EXIF and other image information using an AWS SDK \(p. 1611\)](#)

## Detect PPE in images with Amazon Rekognition using an AWS SDK

The following code examples show how to build an app that uses Amazon Rekognition to detect Personal Protective Equipment (PPE) in images.

Java

### SDK for Java 2.x

Shows how to create an AWS Lambda function that detects images with Personal Protective Equipment.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

### SDK for JavaScript V3

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an application to detect personal protective equipment (PPE) in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app saves the results to an Amazon DynamoDB table, and sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for PPE using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Update a DynamoDB table with results.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

## Detect faces in an image using an AWS SDK

The following code example shows how to:

- Save an image in an Amazon Simple Storage Service (Amazon S3) bucket.
- Use Amazon Rekognition (Amazon Rekognition) to detect facial details, such as age range, gender, and emotion (smiling, etc.).
- Display those details.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

Save the image in an Amazon Simple Storage Service bucket with an **uploads** prefix, use Amazon Rekognition to detect facial details, such as age range, gender, and emotion (smiling, etc.), and display those details.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon S3

## Detect objects in images with Amazon Rekognition using an AWS SDK

The following code examples show how to build an app that uses Amazon Rekognition to detect objects by category in images.

.NET

**AWS SDK for .NET**

Shows how to use Amazon Rekognition .NET API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

**Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

**SDK for Java 2.x**

Shows how to use Amazon Rekognition Java API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

**Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

**SDK for JavaScript V3**

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for objects using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

**Services used in this example**

- Amazon Rekognition

- Amazon S3
- Amazon SES

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to use Amazon Rekognition Kotlin API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

### SDK for Python (Boto3)

Shows you how to use the AWS SDK for Python (Boto3) to create a web application that lets you do the following:

- Upload photos to an Amazon Simple Storage Service (Amazon S3) bucket.
- Use Amazon Rekognition to analyze and label the photos.
- Use Amazon Simple Email Service (Amazon SES) to send email reports of image analysis.

This example contains two main components: a webpage written in JavaScript that is built with React, and a REST service written in Python that is built with Flask-RESTful.

You can use the React webpage to:

- Display a list of images that are stored in your S3 bucket.
- Upload images from your computer to your S3 bucket.
- Display images and labels that identify items that are detected in the image.
- Get a report of all images in your S3 bucket and send an email of the report.

The webpage calls the REST service. The service sends requests to AWS to perform the following actions:

- Get and filter the list of images in your S3 bucket.
- Upload photos to your S3 bucket.
- Use Amazon Rekognition to analyze individual photos and get a list of labels that identify items that are detected in the photo.
- Analyze all photos in your S3 bucket and use Amazon SES to email a report.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Detect people and objects in a video with Amazon Rekognition using an AWS SDK

The following code examples show how to detect people and objects in a video with Amazon Rekognition.

Java

### SDK for Java 2.x

Shows how to use Amazon Rekognition Java API to create an app to detect faces and objects in videos located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

### SDK for JavaScript V3

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an app to detect faces and objects in videos located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for PPE using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition

- Amazon S3
- Amazon SES

Python

#### **SDK for Python (Boto3)**

Use Amazon Rekognition to detect faces, objects, and people in videos by starting asynchronous detection jobs. This example also configures Amazon Rekognition to notify an Amazon Simple Notification Service (Amazon SNS) topic when jobs complete and subscribes an Amazon Simple Queue Service (Amazon SQS) queue to the topic. When the queue receives a message about a job, the job is retrieved and the results are output.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Amazon Rekognition
- Amazon SNS
- Amazon SQS

## **Save EXIF and other image information using an AWS SDK**

The following code example shows how to:

- Get EXIF information from a a JPG, JPEG, or PNG file.
- Upload the image file to an Amazon Simple Storage Service (Amazon S3) bucket.
- Use Amazon Rekognition (Amazon Rekognition) to identify the three top attributes (labels in Amazon Rekognition) in the file.
- Add the EXIF and label information to a Amazon DynamoDB (DynamoDB) table in the Region.

Rust

#### **SDK for Rust**

##### **Note**

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

Get EXIF information from a JPG, JPEG, or PNG file, upload the image file to an Amazon Simple Storage Service bucket, use Amazon Rekognition to identify the three top attributes (*labels* in Amazon Rekognition) in the file, and add the EXIF and label information to a Amazon DynamoDB table in the Region.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- DynamoDB
- Amazon Rekognition
- Amazon S3

# Code examples for Route 53 using AWS SDKs

The following code examples show how to use Amazon Route 53 with an AWS software development kit (SDK).

## Code examples

- [Actions for Route 53 using AWS SDKs \(p. 1612\)](#)
- [Get a list of the public and private Route 53 hosted zones using an AWS SDK \(p. 1612\)](#)

## Actions for Route 53 using AWS SDKs

The following code examples show how to use Amazon Route 53 with AWS SDKs. Each example calls an individual service function.

### Examples

- [Get a list of the public and private Route 53 hosted zones using an AWS SDK \(p. 1612\)](#)

## Get a list of the public and private Route 53 hosted zones using an AWS SDK

The following code example shows how to get a list of the public and private hosted zones.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_host_info(client: &aws_sdk_route53::Client) -> Result<(),  
aws_sdk_route53::Error> {  
    let hosted_zone_count = client.get_hosted_zone_count().send().await?;  
  
    println!(  
        "Number of hosted zones in region : {}",  
        hosted_zone_count.hosted_zone_count().unwrap_or_default(),  
    );  
  
    let hosted_zones = client.list_hosted_zones().send().await?;  
  
    println!("Zones:");  
  
    for hz in hosted_zones.hosted_zones().unwrap_or_default() {  
        let zone_name = hz.name().unwrap_or_default();  
        let zone_id = hz.id().unwrap_or_default();  
  
        println!(" ID : {}", zone_id);  
        println!(" Name : {}", zone_name);  
        println!();  
    }  
}
```

```
    Ok(())  
}
```

- For API details, see [ListHostedZones](#) in *AWS SDK for Rust API reference*.

## Code examples for Route 53 domain registration using AWS SDKs

The following code examples show how to use Amazon Route 53 domain registration with an AWS software development kit (SDK).

### Get started

#### Hello Route 53 domain registration

The following code example shows how to get started using Amazon Route 53 domain registration.

.NET

##### AWS SDK for .NET

###### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static class HelloRoute53Domains  
{  
    static async Task Main(string[] args)  
    {  
        // Use the AWS .NET Core Setup package to set up dependency injection for  
        // the Amazon Route 53 domain registration service.  
        // Use your AWS profile name, or leave it blank to use the default profile.  
        using var host = Host.CreateDefaultBuilder(args)  
            .ConfigureServices(_ =>  
                services.AddAWSService<IAmazonRoute53Domains>()  
            ).Build();  
  
        // Now the client is available for injection.  
        var route53Client =  
            host.Services.GetRequiredService<IAmazonRoute53Domains>();  
  
        // You can use await and any of the async methods to get a response.  
        var response = await route53Client.ListPricesAsync(new ListPricesRequest  
        { Tld = "com" });  
        Console.WriteLine($"Hello Amazon Route 53 Domains! Following are prices  
        for .com domain operations:");  
        var comPrices = response.Prices.FirstOrDefault();  
        if (comPrices != null)  
        {  
            Console.WriteLine($"\\tRegistration:  
            {comPrices.RegistrationPrice?.Price} {comPrices.RegistrationPrice?.Currency}");  
            Console.WriteLine($"\\tRenewal: {comPrices.RenewalPrice?.Price}  
            {comPrices.RenewalPrice?.Currency}");  
        }  
    }  
}
```

- For API details, see [ListPrices](#) in *AWS SDK for .NET API Reference*.

### Code examples

- [Actions for Route 53 domain registration using AWS SDKs \(p. 1614\)](#)
  - [Check the availability of a domain using an AWS SDK \(p. 1614\)](#)
  - [Check the transferability of a domain using an AWS SDK \(p. 1615\)](#)
  - [Get Route 53 domain registration domain details using an AWS SDK \(p. 1615\)](#)
  - [Get details on a Route 53 domain registration operation using an AWS SDK \(p. 1616\)](#)
  - [Get Route 53 domain registration domain name suggestions using an AWS SDK \(p. 1617\)](#)
  - [List Route 53 domain registration domain prices using an AWS SDK \(p. 1618\)](#)
  - [List Route 53 domain registration registered domains using an AWS SDK \(p. 1618\)](#)
  - [List Route 53 domain registration operations using an AWS SDK \(p. 1619\)](#)
  - [Register a domain with Route 53 domain registration using an AWS SDK \(p. 1620\)](#)
  - [View Route 53 domain registration billing records using an AWS SDK \(p. 1621\)](#)
- [Scenarios for Route 53 domain registration using AWS SDKs \(p. 1621\)](#)
  - [Get started with Route 53 domain registration using an AWS SDK \(p. 1622\)](#)

## Actions for Route 53 domain registration using AWS SDKs

The following code examples show how to use Amazon Route 53 domain registration with AWS SDKs. Each example calls an individual service function.

### Examples

- [Check the availability of a domain using an AWS SDK \(p. 1614\)](#)
- [Check the transferability of a domain using an AWS SDK \(p. 1615\)](#)
- [Get Route 53 domain registration domain details using an AWS SDK \(p. 1615\)](#)
- [Get details on a Route 53 domain registration operation using an AWS SDK \(p. 1616\)](#)
- [Get Route 53 domain registration domain name suggestions using an AWS SDK \(p. 1617\)](#)
- [List Route 53 domain registration domain prices using an AWS SDK \(p. 1618\)](#)
- [List Route 53 domain registration registered domains using an AWS SDK \(p. 1618\)](#)
- [List Route 53 domain registration operations using an AWS SDK \(p. 1619\)](#)
- [Register a domain with Route 53 domain registration using an AWS SDK \(p. 1620\)](#)
- [View Route 53 domain registration billing records using an AWS SDK \(p. 1621\)](#)

## Check the availability of a domain using an AWS SDK

The following code example shows how to check the availability of a domain.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Check the availability of a domain name.
/// </summary>
/// <param name="domain">The domain to check for availability.</param>
/// <returns>An availability result string.</returns>
public async Task<string> CheckDomainAvailability(string domain)
{
    var result = await _amazonRoute53Domains.CheckDomainAvailabilityAsync(
        new CheckDomainAvailabilityRequest
    {
        DomainName = domain
    });
    return result.Availability.Value;
}
```

- For API details, see [CheckDomainAvailability](#) in *AWS SDK for .NET API Reference*.

## Check the transferability of a domain using an AWS SDK

The following code example shows how to check the transferability of a domain.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Check the transferability of a domain name.
/// </summary>
/// <param name="domain">The domain to check for transferability.</param>
/// <returns>A transferability result string.</returns>
public async Task<string> CheckDomainTransferability(string domain)
{
    var result = await _amazonRoute53Domains.CheckDomainTransferabilityAsync(
        new CheckDomainTransferabilityRequest
    {
        DomainName = domain
    });
    return result.Transferability.Transferable.Value;
}
```

- For API details, see [CheckDomainTransferability](#) in *AWS SDK for .NET API Reference*.

## Get Route 53 domain registration domain details using an AWS SDK

The following code example shows how to get the details for a domain.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get details for a domain.
/// </summary>
/// <returns>A string with detail information about the domain.</returns>
public async Task<string> GetDomainDetail(string domainName)
{
    try
    {
        var result = await _amazonRoute53Domains.GetDomainDetailAsync(
            new GetDomainDetailRequest()
            {
                DomainName = domainName
            });
        var details = $"\\tDomain {domainName}:\n" +
                     $"\\tCreated on {result.CreationDate.ToShortDateString()}.\n" +
                     $"\\tAdmin contact is {result.AdminContact.Email}.\n" +
                     $"\\tAuto-renew is {result.AutoRenew}.\n";

        return details;
    }
    catch (InvalidArgumentException)
    {
        return $"Domain {domainName} was not found in your account.";
    }
}
```

- For API details, see [GetDomainDetail](#) in *AWS SDK for .NET API Reference*.

## Get details on a Route 53 domain registration operation using an AWS SDK

The following code example shows how to get details on an operation.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get details for a domain action operation.
/// </summary>
/// <param name="operationId">The operational Id.</param>
/// <returns>A string describing the operational details.</returns>
```

```
public async Task<string> GetOperationDetail(string? operationId)
{
    if (operationId == null)
        return "Unable to get operational details because ID is null.";
    try
    {
        var operationDetails =
            await _amazonRoute53Domains.GetOperationDetailAsync(
                new GetOperationDetailRequest
                {
                    OperationId = operationId
                }
            );

        var details = $"\\tOperation {operationId}:\n" +
                     $"\\tFor domain {operationDetails.DomainName} on\n{operationDetails.SubmittedDate.ToShortDateString()}.\\n" +
                     $"\\tMessage is {operationDetails.Message}.\\n" +
                     $"\\tStatus is {operationDetails.Status}.\\n";

        return details;
    }
    catch (AmazonRoute53DomainsException ex)
    {
        return $"Unable to get operation details. Here's why: {ex.Message}.";
    }
}
```

- For API details, see [GetOperationDetail](#) in *AWS SDK for .NET API Reference*.

## Get Route 53 domain registration domain name suggestions using an AWS SDK

The following code example shows how to get domain name suggestions.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get a list of suggestions for a given domain.
/// </summary>
/// <param name="domain">The domain to check for suggestions.</param>
/// <param name="onlyAvailable">If true, only returns available domains.</param>
/// <param name="suggestionCount">The number of suggestions to return. Defaults to the max of 50.</param>
/// <returns>A collection of domain suggestions.</returns>
public async Task<List<DomainSuggestion>> GetDomainSuggestions(string domain,
bool onlyAvailable, int suggestionCount = 50)
{
    var result = await _amazonRoute53Domains.GetDomainSuggestionsAsync(
        new GetDomainSuggestionsRequest
        {
            DomainName = domain,
```

```
        OnlyAvailable = onlyAvailable,
        SuggestionCount = suggestionCount
    }
}
return result.SuggestionsList;
}
```

- For API details, see [GetDomainSuggestions](#) in *AWS SDK for .NET API Reference*.

## List Route 53 domain registration domain prices using an AWS SDK

The following code example shows how to list domain prices.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// List prices for domain type operations.
/// </summary>
/// <param name="domainTypes">Domain types to include in the results.</param>
/// <returns>The list of domain prices.</returns>
public async Task<List<DomainPrice>> ListPrices(List<string> domainTypes)
{
    var results = new List<DomainPrice>();
    var paginatePrices = _amazonRoute53Domains.Paginator.ListPrices(new
ListPricesRequest());
    // Get the entire list using the paginator.
    await foreach (var prices in paginatePrices.Prices)
    {
        results.Add(prices);
    }
    return results.Where(p => domainTypes.Contains(p.Name)).ToList();
}
```

- For API details, see [ListPrices](#) in *AWS SDK for .NET API Reference*.

## List Route 53 domain registration registered domains using an AWS SDK

The following code example shows how to list the registered domains.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// List the domains for the account.
/// </summary>
/// <returns>A collection of domain summary records.</returns>
public async Task<List<DomainSummary>> ListDomains()
{
    var results = new List<DomainSummary>();
    var paginateDomains = _amazonRoute53Domains.Paginator.ListDomains(
        new ListDomainsRequest());

    // Get the entire list using the paginator.
    await foreach (var domain in paginateDomains.Domains)
    {
        results.Add(domain);
    }
    return results;
}
```

- For API details, see [ListDomains in AWS SDK for .NET API Reference](#).

## List Route 53 domain registration operations using an AWS SDK

The following code example shows how to list operations.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// List operations for the account that are submitted after a specified date.
/// </summary>
/// <returns>A collection of operation summary records.</returns>
public async Task<List<OperationSummary>> ListOperations(DateTime
submittedSince)
{
    var results = new List<OperationSummary>();
    var paginateOperations = _amazonRoute53Domains.Paginator.ListOperations(
        new ListOperationsRequest()
    {
        SubmittedSince = submittedSince
    });

    // Get the entire list using the paginator.
    await foreach (var operations in paginateOperations.Operations)
    {
        results.Add(operations);
    }
    return results;
}
```

- For API details, see [ListOperations in AWS SDK for .NET API Reference](#).

## Register a domain with Route 53 domain registration using an AWS SDK

The following code example shows how to register a domain.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Initiate a domain registration request.
/// </summary>
/// <param name="contact">Contact details.</param>
/// <param name="domainName">The domain name to register.</param>
/// <param name="autoRenew">True if the domain should automatically renew.</param>
/// <param name="duration">The duration in years for the domain registration.</param>
/// <returns>The operation Id.</returns>
public async Task<string?> RegisterDomain(string domainName, bool autoRenew,
int duration, ContactDetail contact)
{
    // This example uses the same contact information for admin, registrant,
    and tech contacts.
    try
    {
        var result = await _amazonRoute53Domains.RegisterDomainAsync(
            new RegisterDomainRequest()
            {
                AdminContact = contact,
                RegistrantContact = contact,
                TechContact = contact,
                DomainName = domainName,
                AutoRenew = autoRenew,
                DurationInYears = duration,
                PrivacyProtectAdminContact = false,
                PrivacyProtectRegistrantContact = false,
                PrivacyProtectTechContact = false
            }
        );
        return result.OperationId;
    }
    catch (InvalidArgumentException)
    {
        _logger.LogInformation($"Unable to request registration for domain
{domainName}");
        return null;
    }
}
```

- For API details, see [RegisterDomain](#) in *AWS SDK for .NET API Reference*.

## View Route 53 domain registration billing records using an AWS SDK

The following code example shows how to view billing records.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// View billing records for the account between a start and end date.
/// </summary>
/// <param name="startDate">The start date for billing results.</param>
/// <param name="endDate">The end date for billing results.</param>
/// <returns>A collection of billing records.</returns>
public async Task<List<BillingRecord>> ViewBilling(DateTime startDate, DateTime
endDate)
{
    var results = new List<BillingRecord>();
    var paginateBilling = _amazonRoute53Domains.Paginator.ViewBilling(
        new ViewBillingRequest()
    {
        Start = startDate,
        End = endDate
    });

    // Get the entire list using the paginator.
    await foreach (var billingRecords in paginateBilling.BillingRecords)
    {
        results.Add(billingRecords);
    }
    return results;
}
```

- For API details, see [ViewBilling in AWS SDK for .NET API Reference](#).

## Scenarios for Route 53 domain registration using AWS SDKs

The following code examples show how to use Amazon Route 53 domain registration with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Get started with Route 53 domain registration using an AWS SDK \(p. 1622\)](#)

## Get started with Route 53 domain registration using an AWS SDK

The following code example shows how to:

- List current domains.
- List operations in the past year.
- View billing for the account in the past year.
- View prices for domain types.
- Get domain suggestions.
- Check domain availability.
- Check domain transferability.
- Optionally, request a domain registration.
- Get an operation detail.
- Optionally, get a domain detail.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
public static class Route53DomainScenario
{
    /*
        Before running this .NET code example, set up your development environment,
        including your credentials.

    This .NET example performs the following tasks:
    1. List current domains.
    2. List operations in the past year.
    3. View billing for the account in the past year.
    4. View prices for domain types.
    5. Get domain suggestions.
    6. Check domain availability.
    7. Check domain transferability.
    8. Optionally, request a domain registration.
    9. Get an operation detail.
    10. Optionally, get a domain detail.
    */

    private static Route53Wrapper _route53Wrapper = null!;
    private static IConfiguration _configuration = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
                    LogLevel.Information)
                .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
    }
}
```

```
        .ConfigureServices(_ , services) =>
    services.AddAWSService<IAmazonRoute53Domains>()
        .AddTransient<Route53Wrapper>()
    )
    .Build();

    _configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load settings from .json file.
        .AddJsonFile("settings.local.json",
            true) // Optionally, load local settings.
    .Build();

    var logger = LoggerFactory.Create(builder =>
    {
        builder.AddConsole();
    }).CreateLogger(typeof(Route53DomainScenario));

    _route53Wrapper = host.Services.GetRequiredService<Route53Wrapper>();

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Welcome to the Amazon Route 53 domains example
scenario.");
    Console.WriteLine(new string('-', 80));

    try
    {
        await ListDomains();
        await ListOperations();
        await ListBillingRecords();
        await ListPrices();
        await ListDomainSuggestions();
        await CheckDomainAvailability();
        await CheckDomainTransferability();
        var operationId = await RequestDomainRegistration();
        await GetOperationalDetail(operationId);
        await GetDomainDetails();
    }
    catch (Exception ex)
    {
        logger.LogError(ex, "There was a problem executing the scenario.");
    }

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("The Amazon Route 53 domains example scenario is
complete.");
    Console.WriteLine(new string('-', 80));
}

/// <summary>
/// List account registered domains.
/// </summary>
/// <returns>Async task.</returns>
private static async Task ListDomains()
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"1. List account domains.");
    var domains = await _route53Wrapper.ListDomains();
    for (int i = 0; i < domains.Count; i++)
    {
        Console.WriteLine($"{i + 1}. {domains[i].DomainName}");
    }

    if (!domains.Any())
    {
        Console.WriteLine("\tNo domains found in this account.");
    }
}
```

```
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List domain operations in the past year.
    /// </summary>
    /// <returns>Async task.</returns>
private static async Task ListOperations()
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"2. List account domain operations in the past year.");
    var operations = await _route53Wrapper.ListOperations(
        DateTime.Today.AddYears(-1));
    for (int i = 0; i < operations.Count; i++)
    {
        Console.WriteLine($" \tOperation Id: {operations[i].OperationId}");
        Console.WriteLine($" \tStatus: {operations[i].Status}");
        Console.WriteLine($" \tDate: {operations[i].SubmittedDate}");
    }
    Console.WriteLine(new string('-', 80));
}

    /// <summary>
    /// List billing in the past year.
    /// </summary>
    /// <returns>Async task.</returns>
private static async Task ListBillingRecords()
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"3. View billing for the account in the past year.");
    var billingRecords = await _route53Wrapper.ViewBilling(
        DateTime.Today.AddYears(-1),
        DateTime.Today);
    for (int i = 0; i < billingRecords.Count; i++)
    {
        Console.WriteLine($" \tBill Date:
{billingRecords[i].BillDate.ToShortDateString()}");
        Console.WriteLine($" \tOperation: {billingRecords[i].Operation}");
        Console.WriteLine($" \tPrice: {billingRecords[i].Price}");
    }
    if (!billingRecords.Any())
    {
        Console.WriteLine("\tNo billing records found in this account for the
past year.");
    }
    Console.WriteLine(new string('-', 80));
}

    /// <summary>
    /// List prices for a few domain types.
    /// </summary>
    /// <returns>Async task.</returns>
private static async Task ListPrices()
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"4. View prices for domain types.");
    var domainTypes = new List<string> { "net", "com", "org", "co" };

    var prices = await _route53Wrapper.ListPrices(domainTypes);
    foreach (var pr in prices)
    {
        Console.WriteLine($" \tName: {pr.Name}");
        Console.WriteLine($" \tRegistration: {pr.RegistrationPrice?.Price}
{pr.RegistrationPrice?.Currency}");
    }
}
```

```
Console.WriteLine($"\\tRenewal: {pr.RenewalPrice?.Price}  
{pr.RenewalPrice?.Currency}");  
Console.WriteLine($"\\tTransfer: {pr.TransferPrice?.Price}  
{pr.TransferPrice?.Currency}");  
Console.WriteLine($"\\tChange Ownership:  
{pr.ChangeOwnershipPrice?.Price} {pr.ChangeOwnershipPrice?.Currency}");  
Console.WriteLine($"\\tRestoration: {pr.RestorationPrice?.Price}  
{pr.RestorationPrice?.Currency}");  
Console.WriteLine();  
}  
Console.WriteLine(new string('-', 80));  
}  
  
/// <summary>  
/// List domain suggestions for a domain name.  
/// </summary>  
/// <returns>Async task.</returns>  
private static async Task ListDomainSuggestions()  
{  
    Console.WriteLine(new string('-', 80));  
    Console.WriteLine($"5. Get domain suggestions.");  
    string? domainName = null;  
    while (domainName == null || string.IsNullOrWhiteSpace(domainName))  
    {  
        Console.WriteLine($"Enter a domain name to get available domain  
suggestions.");  
        domainName = Console.ReadLine();  
    }  
  
    var suggestions = await _route53Wrapper.GetDomainSuggestions(domainName,  
true, 5);  
    foreach (var suggestion in suggestions)  
    {  
        Console.WriteLine($"\\tSuggestion Name: {suggestion.DomainName}");  
        Console.WriteLine($"\\tAvailability: {suggestion.Availability}");  
    }  
    Console.WriteLine(new string('-', 80));  
}  
  
/// <summary>  
/// Check availability for a domain name.  
/// </summary>  
/// <returns>Async task.</returns>  
private static async Task CheckDomainAvailability()  
{  
    Console.WriteLine(new string('-', 80));  
    Console.WriteLine($"6. Check domain availability.");  
    string? domainName = null;  
    while (domainName == null || string.IsNullOrWhiteSpace(domainName))  
    {  
        Console.WriteLine($"Enter a domain name to check domain  
availability.");  
        domainName = Console.ReadLine();  
    }  
  
    var availability = await  
_route53Wrapper.CheckDomainAvailability(domainName);  
    Console.WriteLine($"\\tAvailability: {availability}");  
    Console.WriteLine(new string('-', 80));  
}  
  
/// <summary>  
/// Check transferability for a domain name.  
/// </summary>  
/// <returns>Async task.</returns>  
private static async Task CheckDomainTransferability()
```

```
{  
    Console.WriteLine(new string('-', 80));  
    Console.WriteLine($"7. Check domain transferability.");  
    string? domainName = null;  
    while (domainName == null || string.IsNullOrWhiteSpace(domainName))  
    {  
        Console.WriteLine($"Enter a domain name to check domain  
transferability.");  
        domainName = Console.ReadLine();  
    }  
  
    var transferability = await  
_route53Wrapper.CheckDomainTransferability(domainName);  
    Console.WriteLine($"\\tTransferability: {transferability}");  
  
    Console.WriteLine(new string('-', 80));  
}  
  
/// <summary>  
/// Check transferability for a domain name.  
/// </summary>  
/// <returns>Async task.</returns>  
private static async Task<string?> RequestDomainRegistration()  
{  
    Console.WriteLine(new string('-', 80));  
    Console.WriteLine($"8. Optionally, request a domain registration.");  
  
    Console.WriteLine($"\\tNote: This example uses domain request settings in  
settings.json.");  
    Console.WriteLine($"\\tTo change the domain registration settings, set the  
values in that file.");  
    Console.WriteLine($"\\tRemember, registering an actual domain will incur an  
account billing cost.");  
    Console.WriteLine($"\\tWould you like to begin a domain registration? (y/  
n)");  
    var ynResponse = Console.ReadLine();  
    if (ynResponse != null && ynResponse.Equals("y",  
 StringComparison.InvariantCultureIgnoreCase))  
    {  
        string domainName = _configuration["DomainName"];  
        ContactDetail contact = new ContactDetail();  
        contact.CountryCode =  
CountryCode.FindValue(_configuration["Contact:CountryCode"]);  
        contact.ContactType =  
ContactType.FindValue(_configuration["Contact:ContactType"]);  
  
        _configuration.GetSection("Contact").Bind(contact);  
  
        var operationId = await _route53Wrapper.RegisterDomain(  
            domainName,  
            Convert.ToBoolean(_configuration["AutoRenew"]),  
            Convert.ToInt32(_configuration["DurationInYears"]),  
            contact);  
        if (operationId != null)  
        {  
            Console.WriteLine(  
                $"\\tRegistration requested. Operation Id: {operationId}");  
        }  
  
        return operationId;  
    }  
  
    Console.WriteLine(new string('-', 80));  
    return null;  
}
```

```

    ///<summary>
    /// Get details for an operation.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetOperationalDetail(string? operationId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Get an operation detail.");

        var operationDetails =
            await _route53Wrapper.GetOperationDetail(operationId);

        Console.WriteLine(operationDetails);
        Console.WriteLine(new string('-', 80));
    }

    ///<summary>
    /// Optionally, get details for a registered domain.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string?> GetDomainDetails()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Get details on a domain.");

        Console.WriteLine($"\\tNote: you must have a registered domain to get
details.");
        Console.WriteLine($"\\tWould you like to get domain details? (y/n)");
        var ynResponse = Console.ReadLine();
        if (ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase))
        {
            string? domainName = null;
            while (domainName == null)
            {
                Console.WriteLine($"\\tEnter a domain name to get details.");
                domainName = Console.ReadLine();
            }

            var domainDetails = await _route53Wrapper.GetDomainDetail(domainName);
            Console.WriteLine(domainDetails);
        }

        Console.WriteLine(new string('-', 80));
        return null;
    }
}

```

Wrapper methods used by the scenario for Route 53 domain registration actions.

```

public class Route53Wrapper
{
    private readonly IAmazonRoute53Domains _amazonRoute53Domains;
    private readonly ILogger<Route53Wrapper> _logger;
    public Route53Wrapper(IAmazonRoute53Domains amazonRoute53Domains,
ILogger<Route53Wrapper> logger)
    {
        _amazonRoute53Domains = amazonRoute53Domains;
        _logger = logger;
    }
}

```

```
    /// <summary>
    /// List prices for domain type operations.
    /// </summary>
    /// <param name="domainTypes">Domain types to include in the results.</param>
    /// <returns>The list of domain prices.</returns>
    public async Task<List<DomainPrice>> ListPrices(List<string> domainTypes)
    {
        var results = new List<DomainPrice>();
        var paginatePrices = _amazonRoute53Domains.Paginator.ListPrices(new
ListPricesRequest());
        // Get the entire list using the paginator.
        await foreach (var price in paginatePrices.Prices)
        {
            results.Add(price);
        }
        return results.Where(p => domainTypes.Contains(p.Name)).ToList();
    }

    /// <summary>
    /// Check the availability of a domain name.
    /// </summary>
    /// <param name="domain">The domain to check for availability.</param>
    /// <returns>An availability result string.</returns>
    public async Task<string> CheckDomainAvailability(string domain)
    {
        var result = await _amazonRoute53Domains.CheckDomainAvailabilityAsync(
            new CheckDomainAvailabilityRequest
            {
                DomainName = domain
            });
        return result.Availability.Value;
    }

    /// <summary>
    /// Check the transferability of a domain name.
    /// </summary>
    /// <param name="domain">The domain to check for transferability.</param>
    /// <returns>A transferability result string.</returns>
    public async Task<string> CheckDomainTransferability(string domain)
    {
        var result = await _amazonRoute53Domains.CheckDomainTransferabilityAsync(
            new CheckDomainTransferabilityRequest
            {
                DomainName = domain
            });
        return result.Transferability.Transferable.Value;
    }

    /// <summary>
    /// Get a list of suggestions for a given domain.
    /// </summary>
    /// <param name="domain">The domain to check for suggestions.</param>
    /// <param name="onlyAvailable">If true, only returns available domains.</param>
    /// <param name="suggestionCount">The number of suggestions to return. Defaults
    /// to the max of 50.</param>
    /// <returns>A collection of domain suggestions.</returns>
    public async Task<List<DomainSuggestion>> GetDomainSuggestions(string domain,
bool onlyAvailable, int suggestionCount = 50)
    {
        var result = await _amazonRoute53Domains.GetDomainSuggestionsAsync(
```

```

        new GetDomainSuggestionsRequest
    {
        DomainName = domain,
        OnlyAvailable = onlyAvailable,
        SuggestionCount = suggestionCount
    }
);
return result.SuggestionsList;
}

/// <summary>
/// Get details for a domain action operation.
/// </summary>
/// <param name="operationId">The operational Id.</param>
/// <returns>A string describing the operational details.</returns>
public async Task<string> GetOperationDetail(string? operationId)
{
    if (operationId == null)
        return "Unable to get operational details because ID is null.";
    try
    {
        var operationDetails =
            await _amazonRoute53Domains.GetOperationDetailAsync(
                new GetOperationDetailRequest
                {
                    OperationId = operationId
                }
            );

        var details = $"\"\\tOperation {operationId}:\n" +
                     $"\"\\tFor domain {operationDetails.DomainName} on\n" +
{operationDetails.SubmittedDate.ToShortDateString()}.\\n" +
                     $"\"\\tMessage is {operationDetails.Message}.\\n" +
                     $"\"\\tStatus is {operationDetails.Status}.\\n";

        return details;
    }
    catch (AmazonRoute53DomainsException ex)
    {
        return $"Unable to get operation details. Here's why: {ex.Message}.";
    }
}

/// <summary>
/// Initiate a domain registration request.
/// </summary>
/// <param name="contact">Contact details.</param>
/// <param name="domainName">The domain name to register.</param>
/// <param name="autoRenew">True if the domain should automatically renew.</param>
/// <param name="duration">The duration in years for the domain registration.</param>
/// <returns>The operation Id.</returns>
public async Task<string?> RegisterDomain(string domainName, bool autoRenew,
int duration, ContactDetail contact)
{
    // This example uses the same contact information for admin, registrant,
    and tech contacts.
    try
    {
        var result = await _amazonRoute53Domains.RegisterDomainAsync(
            new RegisterDomainRequest()
            {
                AdminContact = contact,

```

```
        RegistrantContact = contact,
        TechContact = contact,
        DomainName = domainName,
        AutoRenew = autoRenew,
        DurationInYears = duration,
        PrivacyProtectAdminContact = false,
        PrivacyProtectRegistrantContact = false,
        PrivacyProtectTechContact = false
    }
);
return result.OperationId;
}
catch (InvalidInputException)
{
    _logger.LogInformation($"Unable to request registration for domain {domainName}");
    return null;
}
}

/// <summary>
/// View billing records for the account between a start and end date.
/// </summary>
/// <param name="startDate">The start date for billing results.</param>
/// <param name="endDate">The end date for billing results.</param>
/// <returns>A collection of billing records.</returns>
public async Task<List<BillingRecord>> ViewBilling(DateTime startDate, DateTime endDate)
{
    var results = new List<BillingRecord>();
    var paginateBilling = _amazonRoute53Domains.Paginator.ViewBilling(
        new ViewBillingRequest()
    {
        Start = startDate,
        End = endDate
    });

    // Get the entire list using the paginator.
    await foreach (var billingRecords in paginateBilling.BillingRecords)
    {
        results.Add(billingRecords);
    }
    return results;
}

/// <summary>
/// List the domains for the account.
/// </summary>
/// <returns>A collection of domain summary records.</returns>
public async Task<List<DomainSummary>> ListDomains()
{
    var results = new List<DomainSummary>();
    var paginateDomains = _amazonRoute53Domains.Paginator.ListDomains(
        new ListDomainsRequest());

    // Get the entire list using the paginator.
    await foreach (var domain in paginateDomains.Domains)
    {
        results.Add(domain);
    }
    return results;
}
```

```

    ///<summary>
    /// List operations for the account that are submitted after a specified date.
    /// </summary>
    /// <returns>A collection of operation summary records.</returns>
    public async Task<List<OperationSummary>> ListOperations(DateTime
submittedSince)
{
    var results = new List<OperationSummary>();
    var paginateOperations = _amazonRoute53Domains.Paginator.ListOperations(
        new ListOperationsRequest()
    {
        SubmittedSince = submittedSince
    });

    // Get the entire list using the paginator.
    await foreach (var operations in paginateOperations.Operations)
    {
        results.Add(operations);
    }
    return results;
}

    ///<summary>
    /// Get details for a domain.
    /// </summary>
    /// <returns>A string with detail information about the domain.</returns>
    public async Task<string> GetDomainDetail(string domainName)
{
    try
    {
        var result = await _amazonRoute53Domains.GetDomainDetailAsync(
            new GetDomainDetailRequest()
        {
            DomainName = domainName
        });
        var details = $"\\tDomain {domainName}:\\n" +
                     $"\\tCreated on {result.CreationDate.ToShortDateString()}\\n" +
                     $"\\tAdmin contact is {result.AdminContact.Email}.\\n" +
                     $"\\tAuto-renew is {result.AutoRenew}.\\n";

        return details;
    }
    catch (InvalidArgumentException)
    {
        return $"Domain {domainName} was not found in your account.";
    }
}
}

```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [CheckDomainAvailability](#)
  - [CheckDomainTransferability](#)
  - [GetDomainDetail](#)
  - [GetDomainSuggestions](#)
  - [GetOperationDetail](#)
  - [ListDomains](#)
  - [ListOperations](#)
  - [ListPrices](#)

- [RegisterDomain](#)
- [ViewBilling](#)

## Code examples for Amazon S3 using AWS SDKs

The following code examples show how to use Amazon Simple Storage Service with an AWS software development kit (SDK).

### Get started

#### Hello Amazon S3

The following code example shows how to get started using Amazon Simple Storage Service (Amazon S3).

##### Python

###### [SDK for Python \(Boto3\)](#)

###### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import boto3

def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) resource and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """
    s3_resource = boto3.resource('s3')
    print("Hello, Amazon S3! Let's list your buckets:")
    for bucket in s3_resource.buckets.all():
        print(f"\t{bucket.name}")

if __name__ == '__main__':
    hello_s3()
```

- For API details, see [ListBuckets in AWS SDK for Python \(Boto3\) API Reference](#).

### Code examples

- [Actions for Amazon S3 using AWS SDKs \(p. 1634\)](#)
  - Add CORS rules to an Amazon S3 bucket using an AWS SDK (p. 1634)
  - Add a lifecycle configuration to an Amazon S3 bucket using an AWS SDK (p. 1639)
  - Add a policy to an Amazon S3 bucket using an AWS SDK (p. 1642)
  - Complete a multipart upload action using an AWS SDK (p. 1647)
  - Copy an object from one Amazon S3 bucket to another using an AWS SDK (p. 1647)
  - Create an Amazon S3 bucket using an AWS SDK (p. 1656)
  - Create a multipart upload structure using an AWS SDK (p. 1664)
  - Delete CORS rules from an Amazon S3 bucket using an AWS SDK (p. 1665)
  - Delete a policy from an Amazon S3 bucket using an AWS SDK (p. 1666)

- Delete an empty Amazon S3 bucket using an AWS SDK (p. 1670)
  - Delete an Amazon S3 object using an AWS SDK (p. 1675)
  - Delete multiple objects from an Amazon S3 bucket using an AWS SDK (p. 1681)
  - Delete the lifecycle configuration of an Amazon S3 bucket using an AWS SDK (p. 1689)
  - Delete the website configuration from an Amazon S3 bucket using an AWS SDK (p. 1690)
  - Determine the existence and content type of an object in an Amazon S3 bucket using an AWS SDK (p. 1692)
  - Determine the existence of an Amazon S3 bucket using an AWS SDK (p. 1693)
  - Get CORS rules for an Amazon S3 bucket using an AWS SDK (p. 1694)
  - Get an object from an Amazon S3 bucket using an AWS SDK (p. 1696)
  - Get the ACL of an Amazon S3 bucket using an AWS SDK (p. 1706)
  - Get the ACL of an Amazon S3 object using an AWS SDK (p. 1709)
  - Get the Region where the Amazon S3 bucket resides using an AWS SDK (p. 1714)
  - Get the lifecycle configuration of an Amazon S3 bucket using an AWS SDK (p. 1715)
  - Get the policy for an Amazon S3 bucket using an AWS SDK (p. 1716)
  - Get the website configuration for an Amazon S3 bucket using an AWS SDK (p. 1719)
  - List Amazon S3 buckets using an AWS SDK (p. 1721)
  - List in-progress multipart uploads to an Amazon S3 bucket using an AWS SDK (p. 1725)
  - List the version of objects in an Amazon S3 bucket using an AWS SDK (p. 1725)
  - List objects in an Amazon S3 bucket using an AWS SDK (p. 1726)
  - Restore an archived copy of an object back into an Amazon S3 bucket using an AWS SDK (p. 1734)
  - Set a new ACL for an Amazon S3 bucket using an AWS SDK (p. 1735)
  - Set the ACL of an Amazon S3 object using an AWS SDK (p. 1738)
  - Set the website configuration for an Amazon S3 bucket using an AWS SDK (p. 1739)
  - Upload a single part of a multipart upload using an AWS SDK (p. 1743)
  - Upload an object to an Amazon S3 bucket using an AWS SDK (p. 1743)
- Scenarios for Amazon S3 using AWS SDKs (p. 1757)
    - Create a presigned URL for Amazon S3 using an AWS SDK (p. 1757)
    - Get started with Amazon S3 buckets and objects using an AWS SDK (p. 1765)
    - Manage versioned Amazon S3 objects in batches with a Lambda function using an AWS SDK (p. 1802)
    - Upload or download large files to and from Amazon S3 using an AWS SDK (p. 1803)
    - Work with Amazon S3 versioned objects using an AWS SDK (p. 1820)
  - Cross-service examples for Amazon S3 using AWS SDKs (p. 1824)
    - Build an Amazon Transcribe app (p. 1824)
    - Convert text to speech and back to text using an AWS SDK (p. 1825)
    - Create a long-lived Amazon EMR cluster and run several steps using an AWS SDK (p. 1825)
    - Create a short-lived Amazon EMR cluster and run a step using an AWS SDK (p. 1826)
    - Create an Amazon Textract explorer application (p. 1826)
    - Detect PPE in images with Amazon Rekognition using an AWS SDK (p. 1827)
    - Detect entities in text extracted from an image using an AWS SDK (p. 1828)
    - Detect faces in an image using an AWS SDK (p. 1829)
  - Detect objects in images with ~~Amazon~~ Rekognition using an AWS SDK (p. 1829)
  - Detect people and objects in a video with Amazon Rekognition using an AWS SDK (p. 1832)
  - Save EXIF and other image information using an AWS SDK (p. 1832)

## Actions for Amazon S3 using AWS SDKs

The following code examples show how to use Amazon Simple Storage Service with AWS SDKs. Each example calls an individual service function.

### Examples

- [Add CORS rules to an Amazon S3 bucket using an AWS SDK \(p. 1634\)](#)
- [Add a lifecycle configuration to an Amazon S3 bucket using an AWS SDK \(p. 1639\)](#)
- [Add a policy to an Amazon S3 bucket using an AWS SDK \(p. 1642\)](#)
- [Complete a multipart upload action using an AWS SDK \(p. 1647\)](#)
- [Copy an object from one Amazon S3 bucket to another using an AWS SDK \(p. 1647\)](#)
- [Create an Amazon S3 bucket using an AWS SDK \(p. 1656\)](#)
- [Create a multipart upload structure using an AWS SDK \(p. 1664\)](#)
- [Delete CORS rules from an Amazon S3 bucket using an AWS SDK \(p. 1665\)](#)
- [Delete a policy from an Amazon S3 bucket using an AWS SDK \(p. 1666\)](#)
- [Delete an empty Amazon S3 bucket using an AWS SDK \(p. 1670\)](#)
- [Delete an Amazon S3 object using an AWS SDK \(p. 1675\)](#)
- [Delete multiple objects from an Amazon S3 bucket using an AWS SDK \(p. 1681\)](#)
- [Delete the lifecycle configuration of an Amazon S3 bucket using an AWS SDK \(p. 1689\)](#)
- [Delete the website configuration from an Amazon S3 bucket using an AWS SDK \(p. 1690\)](#)
- [Determine the existence and content type of an object in an Amazon S3 bucket using an AWS SDK \(p. 1692\)](#)
- [Determine the existence of an Amazon S3 bucket using an AWS SDK \(p. 1693\)](#)
- [Get CORS rules for an Amazon S3 bucket using an AWS SDK \(p. 1694\)](#)
- [Get an object from an Amazon S3 bucket using an AWS SDK \(p. 1696\)](#)
- [Get the ACL of an Amazon S3 bucket using an AWS SDK \(p. 1706\)](#)
- [Get the ACL of an Amazon S3 object using an AWS SDK \(p. 1709\)](#)
- [Get the Region where the Amazon S3 bucket resides using an AWS SDK \(p. 1714\)](#)
- [Get the lifecycle configuration of an Amazon S3 bucket using an AWS SDK \(p. 1715\)](#)
- [Get the policy for an Amazon S3 bucket using an AWS SDK \(p. 1716\)](#)
- [Get the website configuration for an Amazon S3 bucket using an AWS SDK \(p. 1719\)](#)
- [List Amazon S3 buckets using an AWS SDK \(p. 1721\)](#)
- [List in-progress multipart uploads to an Amazon S3 bucket using an AWS SDK \(p. 1725\)](#)
- [List the version of objects in an Amazon S3 bucket using an AWS SDK \(p. 1725\)](#)
- [List objects in an Amazon S3 bucket using an AWS SDK \(p. 1726\)](#)
- [Restore an archived copy of an object back into an Amazon S3 bucket using an AWS SDK \(p. 1734\)](#)
- [Set a new ACL for an Amazon S3 bucket using an AWS SDK \(p. 1735\)](#)
- [Set the ACL of an Amazon S3 object using an AWS SDK \(p. 1738\)](#)
- [Set the website configuration for an Amazon S3 bucket using an AWS SDK \(p. 1739\)](#)
- [Upload a single part of a multipart upload using an AWS SDK \(p. 1743\)](#)
- [Upload an object to an Amazon S3 bucket using an AWS SDK \(p. 1743\)](#)

### Add CORS rules to an Amazon S3 bucket using an AWS SDK

The following code examples show how to add cross-origin resource sharing (CORS) rules to an S3 bucket.

Java

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
    try {
        DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
    .bucket(bucketName)
    .expectedBucketOwner(accountId)
    .build();

        s3.deleteBucketCors(bucketCorsRequest) ;

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {

    try {
        GetBucketCorsRequest bucketCorsRequest = GetBucketCorsRequest.builder()
    .bucket(bucketName)
    .expectedBucketOwner(accountId)
    .build();

        GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
        List<CORSRule> corsRules = corsResponse.corsRules();
        for (CORSRule rule: corsRules) {
            System.out.println("allowOrigins: "+rule.allowedOrigins());
            System.out.println("AllowedMethod: "+rule.allowedMethods());
        }
    } catch (S3Exception e) {

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {

    List<String> allowMethods = new ArrayList<>();
    allowMethods.add("PUT");
    allowMethods.add("POST");
    allowMethods.add("DELETE");

    List<String> allowOrigins = new ArrayList<>();
    allowOrigins.add("http://example.com");
    try {
        // Define CORS rules.
        CORSRule corsRule = CORSRule.builder()
    .allowedMethods(allowMethods)
    .allowedOrigins(allowOrigins)
```

```
.build();

List<CORSRule> corsRules = new ArrayList<>();
corsRules.add(corsRule);
CORSConfiguration configuration = CORSConfiguration.builder()
    .corsRules(corsRules)
    .build();

PutBucketCorsRequest putBucketCorsRequest =
PutBucketCorsRequest.builder()
    .bucket(bucketName)
    .corsConfiguration(configuration)
    .expectedBucketOwner(accountId)
    .build();

s3.putBucketCors(putBucketCorsRequest);

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [PutBucketCors](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Add a CORS rule.

```
// Import required AWS-SDK clients and commands for Node.js.
import { PutBucketCorsCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Set parameters.
// Create initial parameters JSON for putBucketCors.
const thisConfig = {
    AllowedHeaders: ["Authorization"],
    AllowedMethods: [],
    AllowedOrigins: ["*"],
    ExposeHeaders: [],
    MaxAgeSeconds: 3000,
};

// Assemble the list of allowed methods based on command line parameters
```

```
const allowedMethods = [];
process.argv.forEach(function (val) {
  if (val.toUpperCase() === "POST") {
    allowedMethods.push("POST");
  }
  if (val.toUpperCase() === "GET") {
    allowedMethods.push("GET");
  }
  if (val.toUpperCase() === "PUT") {
    allowedMethods.push("PUT");
  }
  if (val.toUpperCase() === "PATCH") {
    allowedMethods.push("PATCH");
  }
  if (val.toUpperCase() === "DELETE") {
    allowedMethods.push("DELETE");
  }
  if (val.toUpperCase() === "HEAD") {
    allowedMethods.push("HEAD");
  }
});
// Copy the array of allowed methods into the config object
thisConfig.AllowMethods = allowedMethods;

// Create an array of configs then add the config object to it.
const corsRules = new Array(thisConfig);

// Create CORS parameters.
export const corsParams = {
  Bucket: "BUCKET_NAME",
  CORSConfiguration: { CORSRules: corsRules },
};
export async function run() {
  try {
    const data = await s3Client.send(new PutBucketCorsCommand(corsParams));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
}
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutBucketCors](#) in [AWS SDK for JavaScript API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
```

```
        that wraps bucket actions in a class-like structure.  
    """  
    self.bucket = bucket  
    self.name = bucket.name  
  
    def put_cors(self, cors_rules):  
        """  
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that  
        are allowed from other domains.  
  
        :param cors_rules: The CORS rules to apply.  
        """  
        try:  
            self.bucket.Cors().put(CORSConfiguration={'CORSRules': cors_rules})  
            logger.info(  
                "Put CORS rules %s for bucket '%s'.", cors_rules, self.bucket.name)  
        except ClientError:  
            logger.exception("Couldn't put CORS rules for bucket %s.",  
self.bucket.name)  
            raise
```

- For API details, see [PutBucketCors](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"  
  
# Wraps Amazon S3 bucket CORS configuration.  
class BucketCorsWrapper  
  attr_reader :bucket_cors  
  
  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with  
  # an existing bucket.  
  def initialize(bucket_cors)  
    @bucket_cors = bucket_cors  
  end  
  
  # Sets CORS rules on a bucket.  
  #  
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.  
  # @param allowed_origins [Array<String>] The origins to allow.  
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.  
  def set_cors(allowed_methods, allowed_origins)  
    @bucket_cors.put(  
      cors_configuration: {  
        cors_rules: [  
          {  
            allowed_methods: allowed_methods,  
            allowed_origins: allowed_origins,  
            allowed_headers: %w[*],  
            max_age_seconds: 3600  
          }  
        ]  
      }  
    )
```

```
)  
    true  
rescue Aws::Errors::ServiceError => e  
    puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:  
#{e.message}"  
    false  
end  
  
end
```

- For API details, see [PutBucketCors](#) in *AWS SDK for Ruby API Reference*.

## Add a lifecycle configuration to an Amazon S3 bucket using an AWS SDK

The following code examples show how to add a lifecycle configuration to an S3 bucket.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void setLifecycleConfig(S3Client s3, String bucketName, String  
accountId) {  
  
    try {  
        // Create a rule to archive objects with the "glacierobjects/" prefix  
        // to Amazon S3 Glacier.  
        LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()  
            .prefix("glacierobjects/")  
            .build();  
  
        Transition transition = Transition.builder()  
            .storageClass(TransitionStorageClass.GLACIER)  
            .days(0)  
            .build();  
  
        LifecycleRule rule1 = LifecycleRule.builder()  
            .id("Archive immediately rule")  
            .filter(ruleFilter)  
            .transitions(transition)  
            .status(ExpirationStatus.ENABLED)  
            .build();  
  
        // Create a second rule.  
        Transition transition2 = Transition.builder()  
            .storageClass(TransitionStorageClass.GLACIER)  
            .days(0)  
            .build();  
  
        List<Transition> transitionList = new ArrayList<>();  
        transitionList.add(transition2);  
  
        LifecycleRuleFilter ruleFilter2 = LifecycleRuleFilter.builder()  
            .prefix("glacierobjects/")  
            .build();
```

```
LifecycleRule rule2 = LifecycleRule.builder()
    .id("Archive and then delete rule")
    .filter(ruleFilter2)
    .transitions(transitionList)
    .status(ExpirationStatus.ENABLED)
    .build();

// Add the LifecycleRule objects to an ArrayList.
ArrayList<LifecycleRule> ruleList = new ArrayList<>();
ruleList.add(rule1);
ruleList.add(rule2);

BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
    .rules(ruleList)
    .build();

PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest =
PutBucketLifecycleConfigurationRequest.builder()
    .bucket(bucketName)
    .lifecycleConfiguration(lifecycleConfiguration)
    .expectedBucketOwner(accountId)
    .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

// Retrieve the configuration and add a new rule.
public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId){

    try {
        GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest =
GetBucketLifecycleConfigurationRequest.builder()
        .bucket(bucketName)
        .expectedBucketOwner(accountId)
        .build();

        GetBucketLifecycleConfigurationResponse response =
s3.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
        List<LifecycleRule> newList = new ArrayList<>();
        List<LifecycleRule> rules = response.rules();
        for (LifecycleRule rule: rules) {
            newList.add(rule);
        }

        // Add a new rule with both a prefix predicate and a tag predicate.
        LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
            .prefix("YearlyDocuments/")
            .build();

        Transition transition = Transition.builder()
            .storageClass(TransitionStorageClass.GLACIER)
            .days(3650)
            .build();

        LifecycleRule rule1 = LifecycleRule.builder()
            .id("NewRule")
```

```
.filter(ruleFilter)
.transitions(transition)
.status(ExpirationStatus.ENABLED)
.build();

// Add the new rule to the list.
newList.add(rule1);
BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
.rules(newList)
.build();

PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest =
PutBucketLifecycleConfigurationRequest.builder()
.bucket(bucketName)
.lifecycleConfiguration(lifecycleConfiguration)
.expectedBucketOwner(accountId)
.build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

} catch (S3Exception e) {
System.err.println(e.awsErrorDetails().errorMessage());
System.exit(1);
}

// Delete the configuration from the Amazon S3 bucket.
public static void deleteLifecycleConfig(S3Client s3, String bucketName, String
accountId) {

try {
DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest.builder()
.bucket(bucketName)
.expectedBucketOwner(accountId)
.build();

s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

} catch (S3Exception e) {
System.err.println(e.awsErrorDetails().errorMessage());
System.exit(1);
}
}
```

- For API details, see [PutBucketLifecycleConfiguration](#) in [AWS SDK for Java 2.x API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
```

```
Boto3 :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        that wraps bucket actions in a class-like structure.
"""
self.bucket = bucket
self.name = bucket.name

def put_lifecycle_configuration(self, lifecycle_rules):
    """
    Apply a lifecycle configuration to the bucket. The lifecycle configuration
    can be used to archive or delete the objects in the bucket according to
    specified parameters, such as a number of days.

    :param lifecycle_rules: The lifecycle rules to apply.
    """
    try:
        self.bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={'Rules': lifecycle_rules})
        logger.info(
            "Put lifecycle rules %s for bucket '%s'.", lifecycle_rules,
            self.bucket.name)
    except ClientError:
        logger.exception(
            "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name)
        raise
```

- For API details, see [PutBucketLifecycleConfiguration](#) in *AWS SDK for Python (Boto3) API Reference*.

## Add a policy to an Amazon S3 bucket using an AWS SDK

The following code examples show how to add a policy to an S3 bucket.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::PutBucketPolicy(const Aws::String &bucketName,
                                  const Aws::String &policyBody,
                                  const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
        Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
        s3_client.PutBucketPolicy(request);
```

```

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: PutBucketPolicy: "
                << outcome.GetError().GetMessage() << std::endl;
        }
        else {
            std::cout << "Set the following policy body for the bucket '" <<
                bucketName << ":" << std::endl << std::endl;
            std::cout << policyBody << std::endl;
        }

        return outcome.IsSuccess();
    }

//! Build a policy JSON string.
/*!
\sa GetPolicyString()
\param userArn Aws user Amazon Resource Name (ARN).
    For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\_identifiers.html#identifiers-arns.
\param bucketName Name of a bucket.
*/
Aws::String GetPolicyString(const Aws::String &userArn,
                           const Aws::String &bucketName) {
    return
        "{\n"
        "  \"Version\": \"2012-10-17\", \n"
        "  \"Statement\": [\n"
        "    {\n"
        "      \"Sid\": \"1\", \n"
        "      \"Effect\": \"Allow\", \n"
        "      \"Principal\": \"\n"
        "      \"AWS\": \""
        + userArn +
        "\", \n"
        "      \"Action\": [ \"s3:GetObject\" ], \n"
        "      \"Resource\": [ \"arn:aws:s3:::"
        + bucketName +
        "/*\" ]\n"
        "    } \n"
        "  ]\n"
        "}";
}

```

- For API details, see [PutBucketPolicy](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

public static void setPolicy(S3Client s3, String bucketName, String policyText)
{
    System.out.println("Setting policy:");
    System.out.println("----");
    System.out.println(policyText);
    System.out.println("----");
}

```

```
System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

try {
    PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
        .bucket(bucketName)
        .policy(policyText)
        .build();

    s3.putBucketPolicy(policyReq);

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Done!");
}

// Loads a JSON-formatted policy from a file
public static String getBucketPolicyFromFile(String policyFile) {

    StringBuilder fileText = new StringBuilder();
    try {
        List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
        for (String line : lines) {
            fileText.append(line);
        }
    }

    } catch (IOException e) {
        System.out.format("Problem reading file: \"%s\"", policyFile);
        System.out.println(e.getMessage());
    }

    try {
        final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
        while (parser.nextToken() != null) {
    }

    } catch (IOException jpe) {
        jpe.printStackTrace();
    }
    return fileText.toString();
}
```

- For API details, see [PutBucketPolicy](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
```

```
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Add the policy.

```
// Import required AWS SDK clients and commands for Node.js.  
import { CreateBucketCommand, PutBucketPolicyCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an  
Amazon S3 service client module.  
  
const BUCKET_NAME = "BUCKET_NAME";  
export const bucketParams = {  
    Bucket: BUCKET_NAME,  
};  
// Create the policy in JSON for the S3 bucket.  
const readOnlyAnonUserPolicy = {  
    Version: "2012-10-17",  
    Statement: [  
        {  
            Sid: "AddPerm",  
            Effect: "Allow",  
            Principal: "*",  
            Action: ["s3:GetObject"],  
            Resource: ["*"],  
        },  
    ],  
};  
  
// Create selected bucket resource string for bucket policy.  
const bucketResource = "arn:aws:s3:::" + BUCKET_NAME + "*"; //BUCKET_NAME  
readOnlyAnonUserPolicy.Statement[0].Resource[0] = bucketResource;  
  
// Convert policy JSON into string and assign into parameters.  
const bucketPolicyParams = {  
    Bucket: BUCKET_NAME,  
    Policy: JSON.stringify(readOnlyAnonUserPolicy),  
};  
  
export const run = async () => {  
    try {  
        const data = await s3Client.send(  
            new CreateBucketCommand(bucketParams)  
        );  
        console.log('Success, bucket created.', data)  
        try {  
            const response = await s3Client.send(  
                new PutBucketPolicyCommand(bucketPolicyParams)  
            );  
            console.log("Success, permissions added to bucket", response);  
            return response;  
        }  
        catch (err) {  
            console.log("Error adding policy to S3 bucket.", err);  
        }  
    } catch (err) {  
        console.log("Error creating S3 bucket.", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [PutBucketPolicy](#) in *AWS SDK for JavaScript API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in  
        Boto3  
                    that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def put_policy(self, policy):  
        """  
        Apply a security policy to the bucket. Policies control users' ability  
        to perform specific actions, such as listing the objects in the bucket.  
        :param policy: The policy to apply to the bucket.  
        """  
        try:  
            self.bucket.Policy().put(Policy=json.dumps(policy))  
            logger.info("Put policy %s for bucket '%s'.", policy, self.bucket.name)  
        except ClientError:  
            logger.exception("Couldn't apply policy to bucket '%s'.",  
                            self.bucket.name)  
            raise
```

- For API details, see [PutBucketPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Wraps an Amazon S3 bucket policy.  
class BucketPolicyWrapper  
  attr_reader :bucket_policy  
  
  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured  
  # with an existing bucket.  
  def initialize(bucket_policy)  
    @bucket_policy = bucket_policy  
  end  
  
  # Sets a policy on a bucket.  
  #  
  def set_policy(policy)
```

```
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
#{e.message}"
    false
  end
end
```

- For API details, see [PutBucketPolicy](#) in *AWS SDK for Ruby API Reference*.

## Complete a multipart upload action using an AWS SDK

The following code example shows how to complete a multipart upload action.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
let _complete_multipart_upload_res = client
  .complete_multipart_upload()
  .bucket(&bucket_name)
  .key(&key)
  .multipart_upload(completed_multipart_upload)
  .upload_id(upload_id)
  .send()
  .await
  .unwrap();
```

- For API details, see [CompleteMultipartUpload](#) in *AWS SDK for Rust API reference*.

## Copy an object from one Amazon S3 bucket to another using an AWS SDK

The following code examples show how to copy an S3 object from one bucket to another.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Copies an object in an Amazon S3 bucket to a folder within the
/// same bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
/// object to copy is located.</param>
/// <param name="objectName">The object to be copied.</param>
/// <param name="folderName">The folder to which the object will
/// be copied.</param>
/// <returns>A boolean value that indicates the success or failure of
/// the copy operation.</returns>
public static async Task<bool> CopyObjectInBucketAsync(
    IAmazonS3 client,
    string bucketName,
    string objectName,
    string folderName)
{
    try
    {
        var request = new CopyObjectRequest
        {
            SourceBucket = bucketName,
            SourceKey = objectName,
            DestinationBucket = bucketName,
            DestinationKey = $"{folderName}\\{objectName}",
        };
        var response = await client.CopyObjectAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error copying object: '{ex.Message}'");
        return false;
    }
}
```

- For API details, see [CopyObject in AWS SDK for .NET API Reference](#).

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::CopyObject(const Aws::String &objectKey, const Aws::String
    &fromBucket, const Aws::String &toBucket,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
        .WithKey(objectKey)
        .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
```

```
        std::cerr << "Error: CopyObject: " <<
            err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
    }
else {
    std::cout << "Successfully copied " << objectKey << " from " << fromBucket
<<
    " to " << toBucket << "." << std::endl;
}
return outcome.IsSuccess();
}
```

- For API details, see [CopyObject](#) in *AWS SDK for C++ API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Copy an object to another name.

// CopyObject is "Pull an object from the source bucket + path".
// The semantics of CopySource varies depending on whether you're using Amazon S3
on Outposts,
// or through access points.
// See https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CopyObject.html#API_CopyObject_RequestSyntax
fmt.Println("Copy an object from another bucket to our bucket.")
_, err = client.CopyObject(context.TODO(), &s3.CopyObjectInput{
    Bucket:      aws.String(name),
    CopySource:  aws.String(name + "/path/myfile.jpg"),
    Key:         aws.String("other/file.jpg"),
})

if err != nil {
    panic("Couldn't copy the object to a new key")
}
```

- For API details, see [CopyObject](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String copyBucketObject (S3Client s3, String fromBucket, String
objectKey, String toBucket) {

    String encodedUrl = "";
    try {
```

```
        encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,  
StandardCharsets.UTF_8.toString());  
  
    } catch (UnsupportedEncodingException e) {  
        System.out.println("URL could not be encoded: " + e.getMessage());  
    }  
  
    CopyObjectRequest copyReq = CopyObjectRequest.builder()  
        .copySourceIfMatch(encodedUrl)  
        .destinationBucket(toBucket)  
        .destinationKey(objectKey)  
        .build();  
  
    try {  
        CopyObjectResponse copyRes = s3.copyObject(copyReq);  
        return copyRes.copyObjectResult().toString();  
    } catch (S3Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [CopyObject](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Copy the object.

```
// Get service clients module and commands using ES6 syntax.  
import { CopyObjectCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js";  
  
// Set the bucket parameters.  
  
export const params = {  
    Bucket: "DESTINATION_BUCKET_NAME",  
    CopySource: "/SOURCE_BUCKET_NAME/OBJECT_NAME",  
    Key: "OBJECT_NAME"  
};  
  
// Create the Amazon S3 bucket.  
export const run = async () => {
```

```
try {
    const data = await s3Client.send(new CopyObjectCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
} catch (err) {
    console.log("Error", err);
}
};

run();
```

- For API details, see [CopyObject in AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String
) {

    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request = CopyObjectRequest {
        copySource = encodedUrl
        bucket = toBucket
        key = objectKey
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- For API details, see [CopyObject in AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Simple copy of an object.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);

try {
    $folder = "copied-folder";
    $s3client->copyObject([
        'Bucket' => $bucket_name,
        'CopySource' => "$bucket_name/$file_name",
        'Key' => "$folder/$file_name-copy",
    ]);
    echo "Copied $file_name to $folder/$file_name-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $file_name with error: " . $exception->getMessage();
    exit("Please fix error with object copying before continuing.");
}
```

- For API details, see [CopyObject in AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
        Boto3
                    that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
        key.
                    This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(CopySource={
                'Bucket': self.object.bucket_name,
                'Key': self.object.key
            })
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name, self.object.key,
                dest_object.bucket_name, dest_object.key)
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name, self.object.key,
                dest_object.bucket_name, dest_object.key)
            raise
```

- For API details, see [CopyObject in AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Copy an object.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
  # used as the source object for
  #                               copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
  # target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
  # nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
#{e.message}"
  end
end

# Replace the source and target bucket names with existing buckets you own and
# replace the source object key
# with an existing object in the source bucket.
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
#{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Copy an object and add server-side encryption to the destination object.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
  # used as the source object for
  #                               copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the
  # target key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
  # nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
    server_side_encryption: encryption)
    target_bucket.object(target_object_key)
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
#{e.message}"
    end
  end

  # Replace the source and target bucket names with existing buckets you own and
  # replace the source object key
  # with an existing object in the source bucket.
  def run_demo
    source_bucket_name = "doc-example-bucket1"
    source_key = "my-source-file.txt"
    target_bucket_name = "doc-example-bucket2"
    target_key = "my-target-file.txt"
    target_encryption = "AES256"

    source_bucket = Aws::S3::Bucket.new(source_bucket_name)
    wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
    target_bucket = Aws::S3::Bucket.new(target_bucket_name)
    target_object = wrapper.copy_object(target_bucket, target_key, target_encryption)
    return unless target_object

    puts "Copied #{source_key} from #{source_bucket_name} to
#{target_object.bucket_name}:#{target_object.key} and \"\
      encrypted the target with #{target_object.server_side_encryption}
      encryption.\""
  end

  run_demo if $PROGRAM_NAME == __FILE__

```

- For API details, see [CopyObject in AWS SDK for Ruby API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn copy_object(
    client: &Client,
    bucket_name: &str,
    object_key: &str,
    target_key: &str,
) -> Result<(), Error> {
    let mut source_bucket_and_object: String = "".to_owned();
    source_bucket_and_object.push_str(bucket_name);
    source_bucket_and_object.push('/');
    source_bucket_and_object.push_str(object_key);

    client
        .copy_object()
        .copy_source(source_bucket_and_object)
        .bucket(bucket_name)
        .key(target_key)
        .send()
        .await?;

    Ok(())
}
```

- For API details, see [CopyObject](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  lo_s3->copyobject(
    iv_bucket = iv_dest_bucket
    iv_key = iv_dest_object
    iv_copiesource = |{ iv_src_bucket }/{ iv_src_object }|
  ).
  MESSAGE 'Object copied to another bucket' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
  MESSAGE 'Object key does not exist' TYPE 'E'.
```

ENDTRY.

- For API details, see [CopyObject](#) in *AWS SDK for SAP ABAP API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func copyFile(from sourceBucket: String, name: String, to destBucket: String) async throws {
    let srcUrl = ("\"sourceBucket)/
\name").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

    let input = CopyObjectInput(
        bucket: destBucket,
        copySource: srcUrl,
        key: name
    )
    - = try await client.copyObject(input: input)
}
```

- For API details, see [CopyObject](#) in *AWS SDK for Swift API reference*.

## Create an Amazon S3 bucket using an AWS SDK

The following code examples show how to create an S3 bucket.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Shows how to create a new Amazon S3 bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>A boolean value representing the success or failure of
/// the bucket creation process.</returns>
public static async Task<bool> CreateBucketAsync(IAmazonS3 client, string
bucketName)
{
    try
{
```

```
        var request = new PutBucketRequest
    {
        BucketName = bucketName,
        UseClientRegion = true,
    };

        var response = await client.PutBucketAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::CreateBucket(const Aws::String &bucketName,
                               const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    //TODO(user): Change the bucket location constraint enum to your target Region.
    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(
            Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: CreateBucket: " <<
                    err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Created bucket " << bucketName <<
                    " in the specified AWS Region." << std::endl;
    }
}

return outcome.IsSuccess();
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Create a bucket: We're going to create a bucket to hold content.
// Best practice is to use the preset private access control list (ACL).
// If you are not creating a bucket from us-east-1, you must specify a bucket
location constraint.
// Bucket names must conform to several rules; read more at https://
docs.aws.amazon.com/AmazonS3/latest/userguide/bucketnamingrules.html
-, err := client.CreateBucket(context.TODO(), &s3.CreateBucketInput{
Bucket:                 aws.String(name),
ACL:                     types.BucketCannedACLPrivate,
CreateBucketConfiguration: &types.CreateBucketConfiguration{LocationConstraint:
types.BucketLocationConstraintUsWest2},
})

if err != nil {
panic("could not create bucket: " + err.Error())
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for Go API Reference*.

Java

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createBucket( S3Client s3Client, String bucketName) {

    try {
        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName +" is ready");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Create the bucket.

```
// Get service clients module and commands using ES6 syntax.  
import { CreateBucketCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js";  
  
// Set the bucket parameters.  
  
export const bucketParams = { Bucket: "BUCKET_NAME" };  
  
// Create the Amazon S3 bucket.  
export const run = async () => {  
    try {  
        const data = await s3Client.send(new CreateBucketCommand(bucketParams));  
        console.log("Success", data);  
        return data; // For unit tests.  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateBucket](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createNewBucket(bucketName: String) {  
  
    val request = CreateBucketRequest {  
        bucket = bucketName  
    }  
  
    S3Client { region = "us-east-1" }.use { s3 ->  
        s3.createBucket(request)  
        println("$bucketName is ready")  
    }  
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);  
  
try {  
    $s3client->createBucket([  
        'Bucket' => $bucket_name,  
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],  
    ]);  
    echo "Created bucket named: $bucket_name \n";  
} catch (Exception $exception) {  
    echo "Failed to create bucket $bucket_name with error: " . $exception->getMessage();  
    exit("Please fix error with bucket creation before continuing.");  
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a bucket with default settings.

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in  
        Boto3
```

```

        that wraps bucket actions in a class-like structure.

"""
self.bucket = bucket
self.name = bucket.name

def create(self, region_override=None):
    """
    Create an Amazon S3 bucket in the default Region for the account or in the
    specified Region.

    :param region_override: The Region in which to create the bucket. If this
    is
                    not specified, the Region configured in your shared
                    credentials is used.
    """
    if region_override is not None:
        region = region_override
    else:
        region = self.bucket.meta.client.meta.region_name
    try:
        self.bucket.create(
            CreateBucketConfiguration={'LocationConstraint': region})

        self.bucket.wait_until_exists()
        logger.info(
            "Created bucket '%s' in region=%s", self.bucket.name, region)
    except ClientError as error:
        logger.exception(
            "Couldn't create bucket named '%s' in region=%s.",
            self.bucket.name, region)
        raise error

```

Create a versioned bucket with a lifecycle configuration.

```

def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
    lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                'LocationConstraint': s3.meta.client.meta.region_name
            })
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response['Error']['Code'] == 'BucketAlreadyOwnedByYou':
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
        bucket = s3.Bucket(bucket_name)

```

```
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

        try:
            bucket.Versioning().enable()
            logger.info("Enabled versioning on bucket %s.", bucket.name)
        except ClientError:
            logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
            raise

        try:
            expiration = 7
            bucket.LifecycleConfiguration().put(
                LifecycleConfiguration={
                    'Rules': [
                        {
                            'Status': 'Enabled',
                            'Prefix': prefix,
                            'NoncurrentVersionExpiration': {'NoncurrentDays': expiration}
                        }
                    ]
                }
            )
            logger.info("Configured lifecycle to expire noncurrent versions after %s days"
days "
                        "on bucket %s.", expiration, bucket.name)
        except ClientError as error:
            logger.warning("Couldn't configure lifecycle on bucket %s because %s. "
                           "Continuing anyway.", bucket.name, error)

    return bucket
```

- For API details, see [CreateBucket](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
  # This is a client-side object until
  #                               create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  end
end
```

```
rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
end

# Gets the Region where the bucket is located.
#
# @return [String] The location of the bucket.
def location
  if @bucket.nil?
    "None. You must create a bucket before you can get its location!"
  else
    @bucket.client.get_bucket_location(bucket: @bucket.name).location_constraint
  end
rescue Aws::Errors::ServiceError => e
  "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
end
end

def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("doc-example-bucket-#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [CreateBucket](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn create_bucket(client: &Client, bucket_name: &str, region: &str) ->
Result<(), Error> {
    let constraint = BucketLocationConstraint::from(region);
    let cfg = CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await?;
    println!("Creating bucket named: {bucket_name}");
    Ok(())
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_s3->createbucket(  
        iv_bucket = iv_bucket_name  
    ).  
    MESSAGE 'S3 bucket created' TYPE 'I'.  
CATCH /aws1/cx_s3_bucketalrdyexists.  
    MESSAGE 'Bucket name already exists' TYPE 'E'.  
CATCH /aws1/cx_s3_bktalrdyownedbyyou.  
    MESSAGE 'Bucket already exist and is owned by you' TYPE 'E'.  
ENDTRY.
```

- For API details, see [CreateBucket](#) in *AWS SDK for SAP ABAP API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func createBucket(name: String) async throws {  
    let config = S3ClientTypes.CreateBucketConfiguration(  
        locationConstraint: .usEast2  
    )  
    let input = CreateBucketInput(  
        bucket: name,  
        createBucketConfiguration: config  
    )  
    _ = try await client.createBucket(input: input)  
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for Swift API reference*.

## Create a multipart upload structure using an AWS SDK

The following code example shows how to create the structure to build a multipart upload action.

Rust

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await
    .unwrap();
```

- For API details, see [CreateMultipartUpload](#) in *AWS SDK for Rust API reference*.

## Delete CORS rules from an Amazon S3 bucket using an AWS SDK

The following code examples show how to delete CORS rules from an S3 bucket.

Python

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                    that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_cors(self):
        """
        Delete the CORS rules from the bucket.

        :param bucket_name: The name of the bucket to update.
        """
        try:
            self.bucket.Cors().delete()
            logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete CORS from bucket '%s'.",
                            self.bucket.name)
```

```
    raise
```

- For API details, see [DeleteBucketCors](#) in *AWS SDK for Python (Boto3) API Reference*.

Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  # an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
#{e.message}"
    false
  end

end
```

- For API details, see [DeleteBucketCors](#) in *AWS SDK for Ruby API Reference*.

## Delete a policy from an Amazon S3 bucket using an AWS SDK

The following code examples show how to delete a policy from an S3 bucket.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::DeleteBucketPolicy(const Aws::String &bucketName,
                                      const Aws::Client::ClientConfiguration
                                      &clientConfig) {
  Aws::S3::S3Client client(clientConfig);
```

```
Aws::S3::Model::DeleteBucketPolicyRequest request;
request.SetBucket(bucketName);

Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
client.DeleteBucketPolicy(request);

if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &err = outcome.GetError();
    std::cerr << "Error: DeleteBucketPolicy: " <<
        err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
}
else {
    std::cout << "Policy was deleted from the bucket." << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [DeleteBucketPolicy](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Delete the bucket policy.
public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {

    DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketPolicy(delReq);
        System.out.println("Done!");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteBucketPolicy](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
```

```
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Delete the bucket policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { DeleteBucketPolicyCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Set the bucket parameters
export const bucketParams = { Bucket: "BUCKET_NAME" };

export const run = async () => {
  try {
    const data = await s3Client.send(new DeleteBucketPolicyCommand(bucketParams));
    console.log("Success", data + ", bucket policy deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Invoke run() so these examples run out of the box.
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteBucketPolicy](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {

    val request = DeleteBucketPolicyRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- For API details, see [DeleteBucketPolicy](#) in [AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in  
        Boto3  
                           that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def delete_policy(self):  
        """  
        Delete the security policy from the bucket.  
        """  
        try:  
            self.bucket.Policy().delete()  
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)  
        except ClientError:  
            logger.exception("Couldn't delete policy for bucket '%s'.",  
self.bucket.name)  
            raise
```

- For API details, see [DeleteBucketPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Wraps an Amazon S3 bucket policy.  
class BucketPolicyWrapper  
  attr_reader :bucket_policy  
  
  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured  
  # with an existing bucket.  
  def initialize(bucket_policy)  
    @bucket_policy = bucket_policy  
  end  
  
  def delete_policy  
    @bucket_policy.delete  
    true  
  rescue Aws::Errors::ServiceError => e  
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's  
why: #{e.message}"  
    false  
  end
```

```
end
```

- For API details, see [DeleteBucketPolicy](#) in *AWS SDK for Ruby API Reference*.

## Delete an empty Amazon S3 bucket using an AWS SDK

The following code examples show how to delete an empty S3 bucket.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Shows how to delete an Amazon S3 bucket.
///</summary>
///<param name="client">An initialized Amazon S3 client object.</param>
///<param name="bucketName">The name of the Amazon S3 bucket to delete.</param>
public static async Task<bool> DeleteBucketAsync(IAmazonS3 client, string bucketName)
{
    var request = new DeleteBucketRequest
    {
        BucketName = bucketName,
    };

    var response = await client.DeleteBucketAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [DeleteBucket](#) in *AWS SDK for .NET API Reference*.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::DeleteBucket(const Aws::String &bucketName,
                               const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketRequest request;
```

```
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: DeleteBucket: " <<
            err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteBucket](#) in *AWS SDK for C++ API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
fmt.Println("Delete a bucket")
// Delete the bucket.

_, err = client.DeleteBucket(context.TODO(), &s3.DeleteBucketInput{
    Bucket: aws.String(name),
})
if err != nil {
    panic("Couldn't delete bucket: " + err.Error())
}
```

- For API details, see [DeleteBucket](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

- For API details, see [DeleteBucket](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Delete the bucket.

```
// Import required AWS SDK clients and commands for Node.js.  
import { DeleteBucketCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an  
Amazon S3 service client module.  
  
// Set the bucket parameters  
export const bucketParams = { Bucket: "BUCKET_NAME" };  
  
export const run = async () => {  
    try {  
        const data = await s3Client.send(new DeleteBucketCommand(bucketParams));  
        return data; // For unit tests.  
        console.log("Success - bucket deleted");  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
// Invoke run() so these examples run out of the box.  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteBucket](#) in [AWS SDK for JavaScript API Reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an empty bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);  
  
try {  
    $s3client->deleteBucket([  
        'Bucket' => $bucket_name,  
    ]);  
}
```

```
        echo "Deleted bucket $bucket_name.\n";
} catch (Exception $exception) {
    echo "Failed to delete $bucket_name with error: " . $exception->getMessage();
    exit("Please fix error with bucket deletion before continuing.");
}
```

- For API details, see [DeleteBucket](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                    that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete(self):
        """
        Delete the bucket. The bucket must be empty or an error is raised.
        """
        try:
            self.bucket.delete()
            self.bucket.wait_until_not_exists()
            logger.info("Bucket %s successfully deleted.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete bucket %s.", self.bucket.name)
            raise
```

- For API details, see [DeleteBucket](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")
    answer = gets.chomp.downcase
    if answer == "y"
```

```
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- For API details, see [DeleteBucket](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_bucket(client: &Client, bucket_name: &str) -> Result<(), Error>
{
    client.delete_bucket().bucket(bucket_name).send().await?;
    println!("Bucket deleted");
    Ok(())
}
```

- For API details, see [DeleteBucket](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.

    lo_s3->deletebucket(
        iv_bucket = iv_bucket_name
    ).
    MESSAGE 'Deleted S3 bucket' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist' TYPE 'E'.
ENDTRY.
```

- For API details, see [DeleteBucket](#) in *AWS SDK for SAP ABAP API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    _ = try await client.deleteBucket(input: input)
}
```

- For API details, see [DeleteBucket](#) in *AWS SDK for Swift API reference*.

## Delete an Amazon S3 object using an AWS SDK

The following code examples show how to delete an S3 object.

### C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::DeleteObject(const Aws::String &objectKey,
                               const Aws::String &fromBucket,
                               const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
        .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: DeleteObject: " <<
            err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted the object." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteObject](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Delete a single object.
fmt.Println("Delete an object from a bucket")
-, err := client.DeleteObject(context.TODO(), &s3.DeleteObjectInput{
    Bucket: aws.String(name),
    Key:    aws.String("other/file.jpg"),
})
if err != nil {
    panic("Couldn't delete object!")
}
```

- For API details, see [DeleteObject](#) in *AWS SDK for Go API Reference*.

JavaScript

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Delete an object.

```
import { DeleteObjectCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js" // Helper function that creates an
Amazon S3 service client module.

export const bucketParams = { Bucket: "BUCKET_NAME", Key: "KEY" };

export const run = async () => {
    try {
        const data = await s3Client.send(new DeleteObjectCommand(bucketParams));
        console.log("Success. Object deleted.", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
```

```
run();
```

- For API details, see [DeleteObject](#) in *AWS SDK for JavaScript API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an object.

```
class ObjectWrapper:  
    """Encapsulates S3 object actions."""  
    def __init__(self, s3_object):  
        """  
        :param s3_object: A Boto3 Object resource. This is a high-level resource in  
        Boto3  
                           that wraps object actions in a class-like structure.  
        """  
        self.object = s3_object  
        self.key = self.object.key  
  
    def delete(self):  
        """  
        Deletes the object.  
        """  
        try:  
            self.object.delete()  
            self.object.wait_until_not_exists()  
            logger.info(  
                "Deleted object '%s' from bucket '%s'.",  
                self.object.key, self.object.bucket_name)  
        except ClientError:  
            logger.exception(  
                "Couldn't delete object '%s' from bucket '%s'.",  
                self.object.key, self.object.bucket_name)  
        raise
```

Roll an object back to a previous version by deleting later versions of the object.

```
def rollback_object(bucket, object_key, version_id):  
    """  
    Rolls back an object to an earlier version by deleting all versions that  
    occurred after the specified rollback version.  
  
    Usage is shown in the usage_demo_single_object function at the end of this  
    module.  
  
    :param bucket: The bucket that holds the object to roll back.  
    :param object_key: The object to roll back.  
    :param version_id: The version ID to roll back to.  
    """  
    # Versions must be sorted by last_modified date because delete markers are  
    # at the end of the list even when they are interspersed in time.  
    versions = sorted(bucket.object_versions.filter(Prefix=object_key),  
                      key=attrgetter('last_modified'), reverse=True)
```

```

logger.debug(
    "Got versions:\n%s",
    '\n'.join([f"\t{version.version_id}, last modified {version.last_modified}"
              for version in versions]))

if version_id in [ver.version_id for ver in versions]:
    print(f"Rolling back to version {version_id}")
    for version in versions:
        if version.version_id != version_id:
            version.delete()
            print(f"Deleted version {version.version_id}")
        else:
            break

    print(f"Active version is now {bucket.Object(object_key).version_id}")
else:
    raise KeyError(f"{version_id} was not found in the list of versions for "
                  f"{object_key}.")

```

Revive a deleted object by removing the object's active delete marker.

```

def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
    marker.
    By removing the delete marker, we make the previous version the latest version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1)

    if 'DeleteMarkers' in response:
        latest_version = response['DeleteMarkers'][0]
        if latest_version['IsLatest']:
            logger.info("Object %s was indeed deleted on %s. Let's revive it.",
                        object_key, latest_version['LastModified'])
            obj = bucket.Object(object_key)
            obj.Version(latest_version['VersionId']).delete()
            logger.info("Revived %s, active version is now %s with body '%s'",
                        object_key, obj.version_id, obj.get()['Body'].read())
        else:
            logger.warning("Delete marker is not the latest version for %s!",
                           object_key)
    elif 'Versions' in response:
        logger.warning("Got an active version for %s, nothing to do.", object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)

```

Create a Lambda handler that removes a delete marker from an S3 object. This handler can be used to efficiently clean up extraneous delete markers in a versioned bucket.

```
import logging
```

```

from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel('INFO')

s3 = boto3.client('s3')


def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
            operation. When the result code is TemporaryFailure, S3 retries the
            operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event['invocationId']
    invocation_schema_version = event['invocationSchemaVersion']

    results = []
    result_code = None
    result_string = None

    task = event['tasks'][0]
    task_id = task['taskId']

    try:
        obj_key = parse.unquote(task['s3Key'], encoding='utf-8')
        obj_version_id = task['s3VersionId']
        bucket_name = task['s3BucketArn'].split(':')[1]

        logger.info("Got task: remove delete marker %s from object %s.",
                   obj_version_id, obj_key)

        try:
            # If this call does not raise an error, the object version is not a
            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id)
            result_code = 'PermanentFailure'
            result_string = f"Object {obj_key}, ID {obj_version_id} is not " \
                           f'a delete marker.'

            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response['ResponseMetadata']['HTTPHeaders'] \
                .get('x-amz-delete-marker', 'false')
            if delete_marker == 'true':
                logger.info("Object %s, version %s is a delete marker.",
                           obj_key, obj_version_id)
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id)
                    result_code = 'Succeeded'
                    result_string = f"Successfully removed delete marker " \
                                   f"{obj_version_id} from object {obj_key}."
                    logger.info(result_string)
                except ClientError as error:
                    logger.error(error)
                    result_code = 'TemporaryFailure'
                    result_string = f"Failed to remove delete marker {obj_version_id} " \
                                   f"from object {obj_key}. Error: {error}"
                    logger.error(result_string)
    except ClientError as error:
        logger.error(error)
        result_code = 'TemporaryFailure'
        result_string = f"Failed to process task {task_id}. Error: {error}"
        logger.error(result_string)

```

```

        # Mark request timeout as a temporary failure so it will be
retried.
        if error.response['Error']['Code'] == 'RequestTimeout':
            result_code = 'TemporaryFailure'
            result_string = f"Attempt to remove delete marker from {obj_key} timed out."
            logger.info(result_string)
        else:
            raise
    else:
        raise ValueError(f"The x-amz-delete-marker header is either not "
                         f"present or is not 'true'.")
except Exception as error:
    # Mark all other exceptions as permanent failures.
    result_code = 'PermanentFailure'
    result_string = str(error)
    logger.exception(error)
finally:
    results.append({
        'taskId': task_id,
        'resultCode': result_code,
        'resultString': result_string
    })
return {
    'invocationSchemaVersion': invocation_schema_version,
    'treatMissingKeysAs': 'PermanentFailure',
    'invocationId': invocation_id,
    'results': results
}

```

- For API details, see [DeleteObject](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

async fn remove_object(client: &Client, bucket: &str, key: &str) -> Result<(), Error> {
    client
        .delete_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await?;

    println!("Object deleted.");
}

Ok(())
}

```

- For API details, see [DeleteObject](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_s3->deleteobject(  
        iv_bucket = iv_bucket_name  
        iv_key = iv_object_key  
    ).  
    MESSAGE 'Object deleted from S3 bucket' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteObject](#) in *AWS SDK for SAP ABAP API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func deleteFile(bucket: String, key: String) async throws {  
    let input = DeleteObjectInput(  
        bucket: bucket,  
        key: key  
    )  
  
    do {  
        _ = try await client.deleteObject(input: input)  
    } catch {  
        throw error  
    }  
}
```

- For API details, see [DeleteObject](#) in *AWS SDK for Swift API reference*.

## Delete multiple objects from an Amazon S3 bucket using an AWS SDK

The following code examples show how to delete multiple objects from an S3 bucket.

.NET

**AWS SDK for .NET**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete all objects in an S3 bucket.

```
/// <summary>
/// Delete all of the objects stored in an existing Amazon S3 bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The name of the bucket from which the
/// contents will be deleted.</param>
/// <returns>A boolean value that represents the success or failure of
/// deleting all of the objects in the bucket.</returns>
public static async Task<bool> DeleteBucketContentsAsync(IAmazonS3 client,
string bucketName)
{
    // Iterate over the contents of the bucket and delete all objects.
    var request = new ListObjectsV2Request
    {
        BucketName = bucketName,
    };

    try
    {
        var response = await client.ListObjectsV2Async(request);

        do
        {
            response.S3Objects
                .ForEach(async obj => await
client.DeleteObjectAsync(bucketName, obj.Key));

            // If the response is truncated, set the request
ContinuationToken
                // from the NextContinuationToken property of the response.
                request.ContinuationToken = response.NextContinuationToken;
        }
        while (response.IsTruncated);

        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting objects: {ex.Message}");
        return false;
    }
}
```

- For API details, see [DeleteObjects](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteBucketObjects(S3Client s3, String bucketName) {

    // Upload three sample objects to the specified Amazon S3 bucket.
    ArrayList<ObjectIdentifier> keys = new ArrayList<>();
    PutObjectRequest putOb;
    ObjectIdentifier objectId;

    for (int i = 0; i < 3; i++) {
        String keyName = "delete object example " + i;
        objectId = ObjectIdentifier.builder()
            .key(keyName)
            .build();

        putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        s3.putObject(putOb, RequestBody.fromString(keyName));
        keys.add(objectId);
    }

    System.out.println(keys.size() + " objects successfully created.");

    // Delete multiple objects in one request.
    Delete del = Delete.builder()
        .objects(keys)
        .build();

    try {
        DeleteObjectsRequest multiObjectDeleteRequest =
DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(del)
            .build();

        s3.deleteObjects(multiObjectDeleteRequest);
        System.out.println("Multiple objects are deleted!");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteObjects](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Delete multiple objects.

```
import { DeleteObjectsCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js" // Helper function that creates an  
Amazon S3 service client module.  
  
export const bucketParams = {  
    Bucket: "BUCKET_NAME",  
    Delete: {  
        Objects: [  
            {  
                Key: "KEY_1",  
            },  
            {  
                Key: "KEY_2",  
            },  
        ],  
    },  
};  
  
export const run = async () => {  
    try {  
        const data = await s3Client.send(new DeleteObjectsCommand(bucketParams));  
        return data; // For unit tests.  
        console.log("Success. Object deleted.");  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

Delete all objects in a bucket.

```
import { ListObjectsCommand, DeleteObjectCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an  
Amazon S3 service client module.  
  
export const bucketParams = { Bucket: "BUCKET_NAME" };  
  
export const run = async () => {  
    try {  
        const data = await s3Client.send(new ListObjectsCommand(bucketParams));  
        let noOfObjects = data.Contents;  
        for (let i = 0; i < noOfObjects.length; i++) {  
            await s3Client.send(  
                new DeleteObjectCommand({  
                    Bucket: bucketParams.Bucket,  
                    Key: noOfObjects[i].Key,  
                })  
            );  
        }  
    }  
};
```

```
        console.log("Success. Objects deleted.");
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
```

- For API details, see [DeleteObjects](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteBucketObjects(bucketName: String, objectName: String) {

    val objectId = ObjectIdentifier {
        key = objectName
    }

    val delOb = Delete {
        objects = listOf(objectId)
    }

    val request = DeleteObjectsRequest {
        bucket = bucketName
        delete = delOb
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}
```

- For API details, see [DeleteObjects](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete a set of objects from a list of keys.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);
```

```
try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [
            'Key' => $content['Key'],
        ];
    }
    $s3client->deleteObjects([
        'Bucket' => $bucket_name,
        'Key' => $file_name,
        'Delete' => [
            'Objects' => $objects,
        ],
    ]);
    $check = $s3client->listObjects([
        'Bucket' => $bucket_name,
    ]);
    if (count($check) <= 0) {
        throw new Exception("Bucket wasn't empty.");
    }
    echo "Deleted all objects and folders from $bucket_name.\n";
} catch (Exception $exception) {
    echo "Failed to delete $file_name from $bucket_name with error: " . $exception->getMessage();
    exit("Please fix error with object deletion before continuing.");
}
```

- For API details, see [DeleteObjects](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete a set of objects by using a list of object keys.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
        Boto3
                    that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    @staticmethod
    def delete_objects(bucket, object_keys):
        """
        Removes a list of objects from a bucket.
        This operation is done as a batch in a single request.

        :param bucket: The bucket that contains the objects. This is a Boto3 Bucket
                       resource.
        :param object_keys: The list of keys that identify the objects to remove.
        :return: The response that contains data about which objects were deleted
                and any that could not be deleted.
        """

```

```

try:
    response = bucket.delete_objects(Delete={
        'Objects': [
            {
                'Key': key
            } for key in object_keys
        ])
    if 'Deleted' in response:
        logger.info(
            "Deleted objects '%s' from bucket '%s'.",
            [del_obj['Key'] for del_obj in response['Deleted']],
            bucket.name)
    if 'Errors' in response:
        logger.warning(
            "Could not delete objects '%s' from bucket '%s'.",
            [{"del_obj['Key']: {del_obj['Code']}"} for del_obj in response['Errors']],
            bucket.name)
except ClientError:
    logger.exception("Couldn't delete any objects from bucket %s.",
                     bucket.name)
    raise
else:
    return response

```

Delete all objects in a bucket.

```

class ObjectWrapper:
    """Encapsulates S3 object actions."""
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
        Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    @staticmethod
    def empty_bucket(bucket):
        """
        Remove all objects from a bucket.

        :param bucket: The bucket to empty. This is a Boto3 Bucket resource.
        """
        try:
            bucket.objects.delete()
            logger.info("Emptied bucket '%s'.", bucket.name)
        except ClientError:
            logger.exception("Couldn't empty bucket '%s'.", bucket.name)
            raise

```

Permanently delete a versioned object by deleting all of its versions.

```

def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.

```

```
:param object_key: The object to delete.  
"""  
try:  
    bucket.object_versions.filter(Prefix=object_key).delete()  
    logger.info("Permanently deleted all versions of object %s.", object_key)  
except ClientError:  
    logger.exception("Couldn't delete all versions of %s.", object_key)  
    raise
```

- For API details, see [DeleteObjects](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.  
#  
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.  
def delete_bucket(bucket)  
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")  
    answer = gets.chomp.downcase  
    if answer == "y"  
        bucket.objects.batch_delete!  
        bucket.delete  
        puts("Emptied and deleted bucket #{bucket.name}.\n")  
    end  
rescue Aws::Errors::ServiceError => e  
    puts("Couldn't empty and delete bucket #{bucket.name}.")  
    puts("\t#{e.code}: #{e.message}")  
    raise  
end
```

- For API details, see [DeleteObjects](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_objects(client: &Client, bucket_name: &str) -> Result<(),  
    Error> {  
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;  
  
    let mut delete_objects: Vec<ObjectIdentifier> = vec![];  
    for obj in objects.contents().unwrap_or_default() {
```

```
        let obj_id = ObjectIdentifier::builder()
            .set_key(Some(obj.key().unwrap().to_string()))
            .build();
        delete_objects.push(obj_id);
    }
    client
        .delete_objects()
        .bucket(bucket_name)
        .delete(Delete::builder().set_objects(Some(delete_objects)).build())
        .send()
        .await?;

    let objects: ListObjectsV2Output =
        client.list_objects_v2().bucket(bucket_name).send().await?;
    match objects.key_count {
        0 => Ok(()),
        _ => Err(Error::Unhandled(Box::from(
            "There were still objects left in the bucket.",
        ))),
    }
}
```

- For API details, see [DeleteObjects](#) in *AWS SDK for Rust API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func deleteObjects(bucket: String, keys: [String]) async throws {
    let input = DeleteObjectsInput(
        bucket: bucket,
        delete: S3ClientTypes.Delete(
            objects: keys.map({ S3ClientTypes.ObjectIdentifier(key: $0) }),
            quiet: true
        )
    )

    do {
        _ = try await client.deleteObjects(input: input)
    } catch {
        throw error
    }
}
```

- For API details, see [DeleteObjects](#) in *AWS SDK for Swift API reference*.

## Delete the lifecycle configuration of an Amazon S3 bucket using an AWS SDK

The following code example shows how to delete the lifecycle configuration of an S3 bucket.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in  
        Boto3  
                    that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def delete_lifecycle_configuration(self):  
        """  
        Remove the lifecycle configuration from the specified bucket.  
        """  
        try:  
            self.bucket.LifecycleConfiguration().delete()  
            logger.info(  
                "Deleted lifecycle configuration for bucket '%s'.",  
                self.bucket.name)  
        except ClientError:  
            logger.exception(  
                "Couldn't delete lifecycle configuration for bucket '%s'.",  
                self.bucket.name)  
        raise
```

- For API details, see [DeleteBucketLifecycle in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete the website configuration from an Amazon S3 bucket using an AWS SDK

The following code examples show how to delete the website configuration from an S3 bucket.

### C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::DeleteBucketWebsite(const Aws::String &bucketName,  
                                     const Aws::Client::ClientConfiguration  
&clientConfig) {  
    Aws::S3::S3Client client(clientConfig);  
    Aws::S3::Model::DeleteBucketWebsiteRequest request;  
    request.SetBucket(bucketName);  
  
    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =  
        client.DeleteBucketWebsite(request);
```

```
if (!outcome.IsSuccess()) {
    auto err = outcome.GetError();
    std::cerr << "Error: DeleteBucketWebsite: " <<
        err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
}
else {
    std::cout << "Website configuration was removed." << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [DeleteBucketWebsite in AWS SDK for C++ API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName) {

    DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketWebsite(delReq);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.out.println("Failed to delete website configuration!");
        System.exit(1);
    }
}
```

- For API details, see [DeleteBucketWebsite in AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Delete the website configuration from the bucket.

```
// Import required AWS SDK clients and commands for Node.js.  
  
import { DeleteBucketWebsiteCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an  
Amazon S3 service client module.  
  
// Create the parameters for calling  
export const bucketParams = { Bucket: "BUCKET_NAME" };  
  
export const run = async () => {  
    try {  
        const data = await s3Client.send(new DeleteBucketWebsiteCommand(bucketParams));  
        return data; // For unit tests.  
        console.log("Success", data);  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteBucketWebsite in AWS SDK for JavaScript API Reference](#).

## Determine the existence and content type of an object in an Amazon S3 bucket using an AWS SDK

The following code examples show how to determine the existence and content type of an object in an S3 bucket.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getContentType (S3Client s3, String bucketName, String  
keyName) {  
  
    try {  
        HeadObjectRequest objectRequest = HeadObjectRequest.builder()  
            .key(keyName)  
            .bucket(bucketName)  
            .build();  
  
        HeadObjectResponse objectHead = s3.headObject(objectRequest);  
        String type = objectHead.contentType();  
        System.out.println("The object content type is "+type);  
    } catch (S3Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
    }  
}
```

```
        System.exit(1);
    }
}
```

- For API details, see [HeadObject in AWS SDK for Java 2.x API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object #{@object.bucket.name}:#{@object.key}."
    puts "Here's why: #{e.message}"
    false
  end
end

# Replace bucket name and object key with an existing bucket and object that you
# own.
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{@object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [HeadObject in AWS SDK for Ruby API Reference](#).

## Determine the existence of an Amazon S3 bucket using an AWS SDK

The following code example shows how to determine the existence of an S3 bucket.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                    that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def exists(self):
        """
        Determine whether the bucket exists and you have access to it.

        :return: True when the bucket exists; otherwise, False.
        """
        try:
            self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
            logger.info("Bucket %s exists.", self.bucket.name)
            exists = True
        except ClientError:
            logger.warning(
                "Bucket %s doesn't exist or you don't have access to it.",
                self.bucket.name)
            exists = False
        return exists
```

- For API details, see [HeadBucket in AWS SDK for Python \(Boto3\) API Reference](#).

## Get CORS rules for an Amazon S3 bucket using an AWS SDK

The following code examples show how to get cross-origin resource sharing (CORS) rules for an S3 bucket.

### JavaScript

#### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Get the CORS policy for the bucket.

```
// Import required AWS SDK clients and commands for Node.js.
import { GetBucketCorsCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Create the parameters for calling
export const bucketParams = { Bucket: "BUCKET_NAME" };

export const run = async () => {
  try {
    const data = await s3Client.send(new GetBucketCorsCommand(bucketParams));
    console.log("Success", JSON.stringify(data.CORSRules));
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetBucketCors](#) in [AWS SDK for JavaScript API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                    that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_cors(self):
        """
        Get the CORS rules for the bucket.

        :return The CORS rules for the specified bucket.
        """
        try:
            cors = self.bucket.Cors()
            logger.info(
                "Got CORS rules %s for bucket '%s'.", cors.cors_rules,
                self.bucket.name)
        except ClientError:
            logger.exception(("Couldn't get CORS for bucket %s.",
                self.bucket.name))
```

```
        raise
    else:
        return cors
```

- For API details, see [GetBucketCors](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  # an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS
  # configuration for the bucket.
  def get_cors
    @bucket_cors.data
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
why: #{e.message}"
      nil
  end
end
```

- For API details, see [GetBucketCors](#) in *AWS SDK for Ruby API Reference*.

## Get an object from an Amazon S3 bucket using an AWS SDK

The following code examples show how to read data from an object in an S3 bucket.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
```

```
/// Shows how to download an object from an Amazon S3 bucket to the
/// local computer.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The name of the bucket where the object is
/// currently stored.</param>
/// <param name="objectName">The name of the object to download.</param>
/// <param name="filePath">The path, including filename, where the
/// downloaded object will be stored.</param>
/// <returns>A boolean value indicating the success or failure of the
/// download process.</returns>
public static async Task<bool> DownloadObjectFromBucketAsync(
    IAmazonS3 client,
    string bucketName,
    string objectName,
    string filePath)
{
    // Create a GetObject request
    var request = new GetObjectRequest
    {
        BucketName = bucketName,
        Key = objectName,
    };

    // Issue request and remember to dispose of the response
    // using GetObjectResponse response = await
    client.GetObjectAsync(request);

    try
    {
        // Save object to local file
        await response.WriteResponseStreamToFileAsync($"{filePath}\\" +
            $"{objectName}", true, System.Threading.CancellationToken.None);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error saving {objectName}: {ex.Message}");
        return false;
    }
}
```

- For API details, see [GetObject](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::GetObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);
```

```
Aws::S3::Model::GetObjectOutcome outcome =
    client.GetObject(request);

if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &err = outcome.GetError();
    std::cerr << "Error: GetObject: " <<
        err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
}
else {
    std::cout << "Successfully retrieved '" << objectKey << "' from ''"
        << fromBucket << '.' << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [GetObject](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Read data as a byte array.

```
public static void getObjectType(S3Client s3, String bucketName, String
keyName, String path) {

    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
        byte[] data = objectBytes.asByteArray();

        // Write the data to a local file.
        File myFile = new File(path );
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
        os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Read tags that belong to an object.

```
public static void listTags(S3Client s3, String bucketName, String keyName) {
```

```
try {
    GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
        .builder()
        .key(keyName)
        .bucket(bucketName)
        .build();

    GetObjectTaggingResponse tags = s3.getObjectTagging(getTaggingRequest);
    List<Tag> tagSet= tags.tagSet();
    for (Tag tag : tagSet) {
        System.out.println(tag.key());
        System.out.println(tag.value());
    }
}

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

Get a URL for an object.

```
public static void getUrl(S3Client s3, String bucketName, String keyName ) {

    try {
        GetUrlRequest request = GetUrlRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        URL url = s3.utilities().getUrl(request);
        System.out.println("The URL for " +keyName + " is " + url);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Get an object by using the S3Presigner client object.

```
public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName ) {

    try {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(60))
            .getObjectRequest(getObjectRequest)
            .build();

        PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
        String theUrl = presignedGetObjectRequest.url().toString();
        System.out.println("Presigned URL: " + theUrl);
    }
}
```

```
HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
    values.forEach(value -> {
        connection.addRequestProperty(header, value);
    });
});

// Send any request payload that the service needs (not needed when
isBrowserExecutable is true).
if (presignedGetObjectRequest.signedPayload().isPresent()) {
    connection.setDoOutput(true);

    try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
        OutputStream httpOutputStream = connection.getOutputStream()) {
        IoUtils.copy(signedPayload, httpOutputStream);
    }
}

// Download the result of executing the request.
try (InputStream content = connection.getInputStream()) {
    System.out.println("Service returned response: ");
    IoUtils.copy(content, System.out);
}

} catch (S3Exception | IOException e) {
    e.printStackTrace();
}
}
```

- For API details, see [GetObject](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Download the object.

```
// Import required AWS SDK clients and commands for Node.js.
import { GetObjectCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

export const bucketParams = {
```

```
    Bucket: "BUCKET_NAME",
    Key: "KEY",
};

export const run = async () => {
  try {
    // Get the object} from the Amazon S3 bucket. It is returned as a
    ReadableStream.
    const data = await s3Client.send(new GetObjectCommand(bucketParams));
    // Convert the ReadableStream to a string.
    return await data.Body.transformToString();
  } catch (err) {
    console.log("Error", err);
  }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetObject](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getObjectBytes(bucketName: String, keyName: String, path: String) {

    val request = GetObjectRequest {
        key = keyName
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- For API details, see [GetObject](#) in [AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get an object.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);

try {
    $file = $s3client->getObject([
        'Bucket' => $bucket_name,
        'Key' => $file_name,
    ]);
    $body = $file->get('Body');
    $body->rewind();
    echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
    echo "Failed to download $file_name from $bucket_name with error: " .
$exception->getMessage();
    exit("Please fix error with file downloading before continuing.");
}
```

- For API details, see [GetObject](#) in *AWS SDK for PHP API Reference*.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
        Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def get(self):
        """
        Gets the object.

        :return: The object data in bytes.
        """
        try:
            body = self.object.get()['Body'].read()
            logger.info(
                "Got object '%s' from bucket '%s'.",
                self.object.key, self.object.bucket_name)
        except ClientError:
            logger.exception(
                "Couldn't get object '%s' from bucket '%s'.",
                self.object.key, self.object.bucket_name)
            raise
        else:
            return body
```

- For API details, see [GetObject](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get an object.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is
  # downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
  # successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
    end
  end

  # Replace bucket name and object key with an existing bucket and object that you
  # own.
  def run_demo
    bucket_name = "doc-example-bucket"
    object_key = "my-object.txt"
    target_path = "my-object-as-file.txt"

    wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
    obj_data = wrapper.get_object(target_path)
    return unless obj_data

    puts "Object #{@object.key} (#{@obj_data.content_length} bytes) downloaded to
    #{target_path}."
  end

  run_demo if $PROGRAM_NAME == __FILE__

```

Get an object and report its server-side encryption state.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end
```

```
end

# Gets the object into memory.
#
# @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
# successful; otherwise nil.
def get_object
  @object.get
rescue Aws::Errors::ServiceError => e
  puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
end
end

# Replace bucket name and object key with an existing bucket and object that you
# own.
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? "no" :
obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [GetObject](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn download_object(client: &Client, bucket_name: &str, key: &str) ->
GetObjectOutput {
  let resp = client
    .get_object()
    .bucket(bucket_name)
    .key(key)
    .send()
    .await;
  resp.unwrap()
}
```

- For API details, see [GetObject](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_s3->getobject(          " oo_result is returned for testing  
purpose "  
        iv_bucket = iv_bucket_name  
        iv_key = iv_object_key  
    ).  
    DATA(lv_object_data) = oo_result->get_body( ).  
    MESSAGE 'Object retrieved from S3 bucket' TYPE 'I'.  
    CATCH /aws1/cx_s3_nosuchbucket.  
        MESSAGE 'Bucket does not exist' TYPE 'E'.  
    CATCH /aws1/cx_s3_nosuchkey.  
        MESSAGE 'Object key does not exist' TYPE 'E'.  
ENDTRY.
```

- For API details, see [GetObject](#) in *AWS SDK for SAP ABAP API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Download an object from a bucket to a local file.

```
public func downloadFile(bucket: String, key: String, to: String) async throws  
{  
    let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)  
  
    let input = GetObjectInput(  
        bucket: bucket,  
        key: key  
    )  
    let output = try await client.getObject(input: input)  
  
    // Get the data stream object. Return immediately if there isn't one.  
    guard let body = output.body else {  
        return  
    }  
    let data = body.toBytes().toData()  
    try data.write(to: fileUrl)  
}
```

Read an object into a Swift Data object.

```
public func readFile(bucket: String, key: String) async throws -> Data {
    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    let output = try await client.getObject(input: input)

    // Get the stream and return its contents in a `Data` object. If
    // there is no stream, return an empty `Data` object instead.
    guard let body = output.body else {
        return "".data(using: .utf8)!
    }
    let data = body.toBytes().toData()
    return data
}
```

- For API details, see [GetObject](#) in *AWS SDK for Swift API reference*.

## Get the ACL of an Amazon S3 bucket using an AWS SDK

The following code examples show how to get the access control list (ACL) of an S3 bucket.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

// S3GetBucketAclAPI defines the interface for the GetBucketAcl function.
// We use this interface to test the function using a mocked service.
type S3GetBucketAclAPI interface {
    GetBucketAcl(ctx context.Context,
        params *s3.GetBucketAclInput,
        optFns ...func(*s3.Options)) (*s3.GetBucketAclOutput, error)
}

// FindBucketAcl retrieves the access control list (ACL) for an Amazon Simple
// Storage Service (Amazon S3) bucket.
// Inputs:
//     c is the context of the method call, which includes the AWS Region
//     api is the interface that defines the method call
//     input defines the input arguments to the service call.
// Output:
//     If success, a GetBucketAclOutput object containing the result of the service
//     call and nil
//     Otherwise, nil and an error from the call to GetBucketAcl
```

```
func FindBucketAcl(c context.Context, api S3GetBucketAclAPI, input
    *s3.GetBucketAclInput) (*s3.GetBucketAclOutput, error) {
    return api.GetBucketAcl(c, input)
}

func main() {
    bucket := flag.String("b", "", "The bucket for which the ACL is returned")
    flag.Parse()

    if *bucket == "" {
        fmt.Println("You must supply a bucket name (-b BUCKET)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := s3.NewFromConfig(cfg)

    input := &s3.GetBucketAclInput{
        Bucket: bucket,
    }

    result, err := FindBucketAcl(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error retrieving ACL for " + *bucket)
        return
    }

    fmt.Println("Owner:", *result.Owner.DisplayName)
    fmt.Println("")
    fmt.Println("Grants")

    for _, g := range result.Grants {
        // If we add a canned ACL, the name is nil
        if g.Grantee.DisplayName == nil {
            fmt.Println(" Grantee: EVERYONE")
        } else {
            fmt.Println(" Grantee: ", *g.Grantee.DisplayName)
        }

        fmt.Println(" Type: ", string(g.Grantee.Type))
        fmt.Println(" Permission: ", string(g.Permission))
        fmt.Println("")
    }
}
```

- For API details, see [GetBucketAcl](#) in [AWS SDK for Go API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String getBucketACL(S3Client s3, String objectKey, String
    bucketName) {
```

```
try {
    GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .build();

    GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
    List<Grant> grants = aclRes.grants();
    String grantee = "";
    for (Grant grant : grants) {
        System.out.format(" %s: %s\n", grant.grantee().id(),
    grant.permission());
        grantee = grant.grantee().id();
    }

    return grantee;
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
```

- For API details, see [GetBucketAcl](#) in [AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Get the ACL permissions.

```
// Import required AWS SDK clients and commands for Node.js.
import { GetBucketAclCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Create the parameters.
export const bucketParams = { Bucket: "BUCKET_NAME" };

export const run = async () => {
    try {
        const data = await s3Client.send(new GetBucketAclCommand(bucketParams));
        console.log("Success", data.Grants);
        return data; // For unit tests.
    }
```

```
        } catch (err) {
            console.log("Error", err);
        }
    };
    run();
}
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetBucketAcl](#) in [AWS SDK for JavaScript API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                           that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_acl(self):
        """
        Get the ACL of the bucket.

        :return: The ACL of the bucket.
        """
        try:
            acl = self.bucket.Acl()
            logger.info(
                "Got ACL for bucket %s. Owner is %s.", self.bucket.name, acl.owner)
        except ClientError:
            logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)
            raise
        else:
            return acl
```

- For API details, see [GetBucketAcl](#) in [AWS SDK for Python \(Boto3\) API Reference](#).

## Get the ACL of an Amazon S3 object using an AWS SDK

The following code examples show how to get the access control list (ACL) of an S3 object.

### C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::GetBucketAcl(const Aws::String &bucketName,
                               const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
        s3_client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: GetBucketAcl: "
              << err.GetExceptionName() << ":" << err.GetMessage() <<
        std::endl;
    }
    else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            std::cout << "For bucket " << bucketName << ": "
                  << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:      "
                      << GetGranteeTypeString(grantee.GetType()) << std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name: "
                      << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                      << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:      "
                      << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:      "
                      << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:   "
                  << GetPermissionString(grant.GetPermission()) <<
            std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
```

```
\sa GetGranteeTypeString()
\param type Type enumeration.
*/
Aws::String GetGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!!
\sa GetPermissionString()
\param permission Permission enumeration.
*/
Aws::String GetPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                   "objects in this bucket, and read/write this "
                   "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }
}

return "Permission unknown";
}
```

- For API details, see [GetObjectAcl](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
```

```

"context"
"flag"
"fmt"

"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/s3"
)

// S3GetObjectAclAPI defines the interface for the GetObjectAcl function.
// We use this interface to test the function using a mocked service.
type S3GetObjectAclAPI interface {
    GetObjectAcl(ctx context.Context,
        params *s3.GetObjectAclInput,
        optFns ...func(*s3.Options)) (*s3.GetObjectAclOutput, error)
}

// FindGetObjectAcl gets the access control list (ACL) for an Amazon Simple Storage
// Service (Amazon S3) bucket object
// Inputs:
//     c is the context of the method call, which includes the AWS Region
//     api is the interface that defines the method call
//     input defines the input arguments to the service call.
// Output:
//     If success, a GetObjectAclOutput object containing the result of the service
//     call and nil
//     Otherwise, nil and an error from the call to GetObjectAcl
func FindGetObjectAcl(c context.Context, api S3GetObjectAclAPI, input
    *s3.GetObjectAclInput) (*s3.GetObjectAclOutput, error) {
    return api.GetObjectAcl(c, input)
}

func main() {
    bucket := flag.String("b", "", "The bucket containing the object")
    objectName := flag.String("o", "", "The bucket object to get ACL from")
    flag.Parse()

    if *bucket == "" || *objectName == "" {
        fmt.Println("You must supply a bucket (-b BUCKET) and object (-o OBJECT)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := s3.NewFromConfig(cfg)

    input := &s3.GetObjectAclInput{
        Bucket: bucket,
        Key:    objectName,
    }

    result, err := FindGetObjectAcl(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error getting ACL for " + *objectName)
        return
    }

    fmt.Println("Owner:", *result.Owner.DisplayName)
    fmt.Println("")
    fmt.Println("Grants")

    for _, g := range result.Grants {
        fmt.Println("  Grantee: ", *g.Grantee.DisplayName)
        fmt.Println("  Type:      ", string(g.Grantee.Type))
    }
}

```

```
    fmt.Println("  Permission:", string(g.Permission))
    fmt.Println(""))
}
```

- For API details, see [GetObjectAcl](#) in *AWS SDK for Go API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getBucketACL(objectKey: String, bucketName: String) {

    val request = GetObjectAclRequest {
        bucket = bucketName
        key = objectKey
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- For API details, see [GetObjectAcl](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
        Boto3
                                         that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def get_acl(self):
        """
```

```
Gets the ACL of the object.

:return: The ACL of the object.
"""
try:
    acl = self.object.Acl()
    logger.info(
        "Got ACL for object %s owned by %s.",
        self.object.key, acl.owner['DisplayName'])
except ClientError:
    logger.exception("Couldn't get ACL for object %s.", self.object.key)
    raise
else:
    return acl
```

- For API details, see [GetObjectAcl](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get the Region where the Amazon S3 bucket resides using an AWS SDK

The following code example shows how to get the Region location for an S3 bucket.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_buckets(strict: bool, client: &Client, region: &str) -> Result<(), Error> {
    let resp = client.list_buckets().send().await?;
    let buckets = resp.buckets().unwrap_or_default();
    let num_buckets = buckets.len();

    let mut in_region = 0;

    for bucket in buckets {
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name()).unwrap_or_default()
                .send()
                .await?;

            if r.location_constraint().unwrap().as_ref() == region {
                println!("{} {}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{} {}", bucket.name().unwrap_or_default());
        }
    }

    println!();
```

```
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            num_buckets,
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- For API details, see [GetBucketLocation](#) in *AWS SDK for Rust API reference*.

## Get the lifecycle configuration of an Amazon S3 bucket using an AWS SDK

The following code example shows how to get the lifecycle configuration of an S3 bucket.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                    that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_lifecycle_configuration(self):
        """
        Get the lifecycle configuration of the bucket.

        :return: The lifecycle rules of the specified bucket.
        """
        try:
            config = self.bucket.LifecycleConfiguration()
            logger.info(
                "Got lifecycle rules %s for bucket '%s'.", config.rules,
                self.bucket.name)
        except:
            logger.exception(
                "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name)
            raise
        else:
            return config.rules
```

- For API details, see [GetBucketLifecycleConfiguration](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get the policy for an Amazon S3 bucket using an AWS SDK

The following code examples show how to get the policy for an S3 bucket.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::GetBucketPolicy(const Aws::String &bucketName,
                                  const Aws::Client::ClientConfiguration
                                  &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
        s3_client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: GetBucketPolicy: "
              << err.GetExceptionName() << ":" << err.GetMessage() <<
        std::endl;
    }
    else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Retrieve the policy for bucket '" << bucketName << "' :\n\n"
        <<
        policy_stream.str() << std::endl;
    }
}

return outcome.IsSuccess();
}
```

- For API details, see [GetBucketPolicy](#) in *AWS SDK for C++ API Reference*.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String getPolicy(S3Client s3, String bucketName) {

    String policyText;
    System.out.format("Getting policy for bucket: \"%s\"\n", bucketName);
    GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
```

```
.bucket(bucketName)
.build();

try {
    GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
    policyText = policyRes.policy();
    return policyText;
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
```

- For API details, see [GetBucketPolicy](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Get the bucket policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { GetBucketPolicyCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Create the parameters for calling
export const bucketParams = { Bucket: "BUCKET_NAME" };

export const run = async () => {
    try {
        const data = await s3Client.send(new GetBucketPolicyCommand(bucketParams));
        console.log("Success", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetBucketPolicy](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getPolicy(bucketName: String): String? {  
    println("Getting policy for bucket $bucketName")  
  
    val request = GetBucketPolicyRequest {  
        bucket = bucketName  
    }  
  
    S3Client { region = "us-east-1" }.use { s3 ->  
        val policyRes = s3.getBucketPolicy(request)  
        return policyRes.policy  
    }  
}
```

- For API details, see [GetBucketPolicy](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in  
        Boto3  
                           that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def get_policy(self):  
        """  
        Get the security policy of the bucket.  
  
        :return: The security policy of the specified bucket, in JSON format.  
        """  
        try:  
            policy = self.bucket.Policy()  
            logger.info("Got policy %s for bucket '%s'.", policy.policy,  
self.bucket.name)  
        except ClientError:  
            logger.exception("Couldn't get policy for bucket '%s'.",  
self.bucket.name)  
            raise
```

```
        else:  
            return json.loads(policy.policy)
```

- For API details, see [GetBucketPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Wraps an Amazon S3 bucket policy.  
class BucketPolicyWrapper  
    attr_reader :bucket_policy  
  
    # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured  
    # with an existing bucket.  
    def initialize(bucket_policy)  
        @bucket_policy = bucket_policy  
    end  
  
    # Gets the policy of a bucket.  
    #  
    # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.  
    def get_policy  
        policy = @bucket_policy.data.policy  
        policy.respond_to?(:read) ? policy.read : policy  
        rescue Aws::Errors::ServiceError => e  
            puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:  
#{e.message}"  
            nil  
        end  
    end
```

- For API details, see [GetBucketPolicy](#) in *AWS SDK for Ruby API Reference*.

## Get the website configuration for an Amazon S3 bucket using an AWS SDK

The following code examples show how to get the website configuration for an S3 bucket.

### C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::GetWebsiteConfig(const Aws::String &bucketName,  
                                    const Aws::Client::ClientConfiguration  
&clientConfig) {  
    Aws::S3::S3Client s3_client(clientConfig);
```

```
Aws::S3::Model::GetBucketWebsiteRequest request;
request.SetBucket(bucketName);

Aws::S3::Model::GetBucketWebsiteOutcome outcome =
    s3_client.GetBucketWebsite(request);

if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &err = outcome.GetError();

    std::cerr << "Error: GetBucketWebsite: "
        << err.GetMessage() << std::endl;
}
else {
    Aws::S3::Model::GetBucketWebsiteResult websiteResult = outcome.GetResult();

    std::cout << "Success: GetBucketWebsite: "
        << std::endl << std::endl
        << "For bucket '" << bucketName << ":"
        << std::endl
        << "Index page : "
        << websiteResult.GetIndexDocument().GetSuffix()
        << std::endl
        << "Error page: "
        << websiteResult.GetErrorDocument().GetKey()
        << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [GetBucketWebsite](#) in *AWS SDK for C++ API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Get the website configuration.

```
// Import required AWS SDK clients and commands for Node.js.
import { GetBucketWebsiteCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Create the parameters for calling
```

```
export const bucketParams = { Bucket: "BUCKET_NAME" };

export const run = async () => {
  try {
    const data = await s3Client.send(new GetBucketWebsiteCommand(bucketParams));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};

run();
```

- For API details, see [GetBucketWebsite](#) in *AWS SDK for JavaScript API Reference*.

## List Amazon S3 buckets using an AWS SDK

The following code examples show how to list S3 buckets.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::ListBuckets(const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    }
    else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << " "
        buckets\n";
        for (auto &b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }

    return result;
}
```

- For API details, see [ListBuckets](#) in *AWS SDK for C++ API Reference*.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
listBucketsResult, err := client.ListBuckets(context.TODO(),
&s3.ListBucketsInput{})

if err != nil {
    panic("Couldn't list buckets")
}

for _, bucket := range listBucketsResult.Buckets {
    fmt.Printf("Bucket name: %s\t\tcreated at: %v\n", *bucket.Name,
bucket.CreationDate)
}
```

- For API details, see [ListBuckets in AWS SDK for Go API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

List the buckets.

```
// Import required AWS SDK clients and commands for Node.js.
import { ListBucketsCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

export const run = async () => {
    try {
        const data = await s3Client.send(new ListBucketsCommand({}));
        console.log("Success", data.Buckets);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListBuckets in AWS SDK for JavaScript API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in  
        Boto3  
                           that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    @staticmethod  
    def list(s3_resource):  
        """  
        Get the buckets in all Regions for the current account.  
  
        :param s3_resource: A Boto3 S3 resource. This is a high-level resource in  
        Boto3  
                           that contains collections and factory methods to create  
                           other high-level S3 sub-resources.  
        :return: The list of buckets.  
        """  
        try:  
            buckets = list(s3_resource.buckets.all())  
            logger.info("Got buckets: %s.", buckets)  
        except ClientError:  
            logger.exception("Couldn't get buckets.")  
            raise  
        else:  
            return buckets
```

- For API details, see [ListBuckets in AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"  
  
# Wraps Amazon S3 resource actions.  
class BucketListWrapper  
  attr_reader :s3_resource  
  
  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.  
  def initialize(s3_resource)  
    @s3_resource = s3_resource  
  end  
  
  # Lists buckets for the current account.
```

```
#  
# @param count [Integer] The maximum number of buckets to list.  
def list_buckets(count)  
    puts "Found these buckets:"  
    @s3_resource.buckets.each do |bucket|  
        puts "\t#{bucket.name}"  
        count -= 1  
        break if count.zero?  
    end  
    true  
rescue Aws::Errors::ServiceError => e  
    puts "Couldn't list buckets. Here's why: #{e.message}"  
    false  
end  
end  
  
def run_demo  
    wrapper = BucketListWrapper.new(Aws::S3::Resource.new)  
    wrapper.list_buckets(25)  
end  
  
run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [ListBuckets in AWS SDK for Ruby API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_buckets(strict: bool, client: &Client, region: &str) -> Result<(), Error> {  
    let resp = client.list_buckets().send().await?;  
    let buckets = resp.buckets().unwrap_or_default();  
    let num_buckets = buckets.len();  
  
    let mut in_region = 0;  
  
    for bucket in buckets {  
        if strict {  
            let r = client  
                .get_bucket_location()  
                .bucket(bucket.name()).unwrap_or_default()  
                .send()  
                .await?;  
  
            if r.location_constraint().unwrap().as_ref() == region {  
                println!("{}: {},", bucket.name().unwrap_or_default());  
                in_region += 1;  
            }  
        } else {  
            println!("{}: {},", bucket.name().unwrap_or_default());  
        }  
    }  
}
```

```
    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- For API details, see [ListBuckets in AWS SDK for Rust API reference](#).

## List in-progress multipart uploads to an Amazon S3 bucket using an AWS SDK

The following code example shows how to list in-progress multipart uploads to an S3 bucket.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listUploads( S3Client s3, String bucketName ) {

    try {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

        ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
        for (MultipartUpload upload: uploads) {
            System.out.println("Upload in progress: Key = \\" + upload.key() +
"\\", id = " + upload.uploadId());
        }
    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListMultipartUploads in AWS SDK for Java 2.x API Reference](#).

## List the version of objects in an Amazon S3 bucket using an AWS SDK

The following code example shows how to list object versions in an S3 bucket.

Rust

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {
    let resp = client.list_object_versions().bucket(bucket).send().await?;

    for version in resp.versions().unwrap_or_default() {
        println!("{}: {}", version.key().unwrap_or_default(), version.version_id().unwrap_or_default());
        println!("version ID: {}", version.version_id().unwrap_or_default());
        println!();
    }

    Ok(())
}
```

- For API details, see [ListObjectVersions](#) in *AWS SDK for Rust API reference*.

## List objects in an Amazon S3 bucket using an AWS SDK

The following code examples show how to list objects in an S3 bucket.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Shows how to list the objects in an Amazon S3 bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The name of the bucket for which to list
/// the contents.</param>
/// <returns>A boolean value indicating the success or failure of the
/// copy operation.</returns>
public static async Task<bool> ListBucketContentsAsync(IAmazonS3 client,
string bucketName)
{
    try
    {
        var request = new ListObjectsV2Request
        {
            BucketName = bucketName,
            MaxKeys = 5,
        };

        Console.WriteLine("-----");
        Console.WriteLine($"Listing the contents of {bucketName}:");
```

```
Console.WriteLine("-----");
var response = new ListObjectsV2Response();
do
{
    response = await client.ListObjectsV2Async(request);

    response.S3Objects
        .ForEach(obj => Console.WriteLine($"{obj.Key},-{35}
{obj.LastModified.ToShortDateString()},10){obj.Size},10}"));

    // If the response is truncated, set the request
ContinuationToken
    // from the NextContinuationToken property of the response.
    request.ContinuationToken = response.NextContinuationToken;
}
while (response.IsTruncated);

return true;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error encountered on server.
Message:{ex.Message}' getting list of objects.");
    return false;
}
}
```

- For API details, see [ListObjects in AWS SDK for .NET API Reference](#).

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::ListObjects(const Aws::String &bucketName,
                             const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::ListObjectsRequest request;
    request.WithBucket(bucketName);

    auto outcome = s3_client.ListObjects(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ListObjects: " <<
                    outcome.GetError().GetMessage() << std::endl;
    }
    else {
        Aws::Vector<Aws::S3::Model::Object> objects =
            outcome.GetResult().GetContents();

        for (Aws::S3::Model::Object &object: objects) {
            std::cout << object.GetKey() << std::endl;
        }
    }
}
```

```
        return outcome.IsSuccess();
    }
```

- For API details, see [ListObjects in AWS SDK for C++ API Reference](#).

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// List objects in the bucket.
// n.b. object keys in Amazon S3 do not begin with '/'. You do not need to lead
your
// prefix with it.
fmt.Println("Listing the objects in the bucket:")
listObjsResponse, err := client.ListObjectsV2(context.TODO(),
&s3.ListObjectsV2Input{
    Bucket: aws.String(name),
    Prefix: aws.String(""),
})

if err != nil {
    panic("Couldn't list bucket contents")
}

for _, object := range listObjsResponse.Contents {
    fmt.Printf("%s (%d bytes, class %v) \n", *object.Key, object.Size,
object.StorageClass)
}
```

- For API details, see [ListObjects in AWS SDK for Go API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listBucketObjects(S3Client s3, String bucketName ) {

    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.print("\n The name of the key is " + myValue.key());
            System.out.print("\n The object is " + calKb(myValue.size()) + " "
KBs");
    }
}
```

```
        System.out.print("\n The owner is " + myValue.owner());
    }

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

//convert bytes to kbs.
private static long calKb(Long val) {
    return val/1024;
}
```

- For API details, see [ListObjects in AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

List the objects.

```
// Import required AWS SDK clients and commands for Node.js.
import { ListObjectsCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Create the parameters for the bucket
export const bucketParams = { Bucket: "BUCKET_NAME" };

export const run = async () => {
    try {
        const data = await s3Client.send(new ListObjectsCommand(bucketParams));
        console.log("Success", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

List 1000 or more objects.

```
// Import required AWS SDK clients and commands for Node.js.
```

```
import { ListObjectsCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Create the parameters for the bucket
export const bucketParams = { Bucket: "BUCKET_NAME" };

export async function run() {
    // Declare truncated as a flag that the while loop is based on.
    let truncated = true;
    // Declare a variable to which the key of the last element is assigned to in the
    response.
    let pageMarker;
    // while loop that runs until 'response.truncated' is false.
    while (truncated) {
        try {
            const response = await s3Client.send(new ListObjectsCommand(bucketParams));
            // return response; //For unit tests
            response.Contents.forEach((item) => {
                console.log(item.Key);
            });
            // Log the key of every item in the response to standard output.
            truncated = response.IsTruncated;
            // If truncated is true, assign the key of the last element in the response
            to the pageMarker variable.
            if (truncated) {
                pageMarker = response.Contents.slice(-1)[0].Key;
                // Assign the pageMarker value to bucketParams so that the next iteration
                starts from the new pageMarker.
                bucketParams.Marker = pageMarker;
            }
            // At end of the list, response.truncated is false, and the function exits
            the while loop.
        } catch (err) {
            console.log("Error", err);
            truncated = false;
        }
    }
}
run();
```

- For API details, see [ListObjects in AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listBucketObjects(bucketName: String) {

    val request = ListObjectsRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
```

```
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${calKb(myObject.size)} KBs")
            println("The owner is ${myObject.owner}")
        }
    }

private fun calKb(intValue: Long): Long {
    return intValue / 1024
}
```

- For API details, see [ListObjects in AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List objects in a bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);

try {
    $contents = $s3client->listObjects([
        'Bucket' => $bucket_name,
    ]);
    echo "The contents of your bucket are: \n";
    foreach ($contents['Contents'] as $content) {
        echo $content['Key'] . "\n";
    }
} catch (Exception $exception) {
    echo "Failed to list objects in $bucket_name with error: " . $exception-
>getMessage();
    exit("Please fix error with listing objects before continuing.");
}
```

- For API details, see [ListObjects in AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
        Boto3
                                         that wraps object actions in a class-like structure.
```

```
"""
    self.object = s3_object
    self.key = self.object.key

@staticmethod
def list(bucket, prefix=None):
    """
        Lists the objects in a bucket, optionally filtered by a prefix.

    :param bucket: The bucket to query. This is a Boto3 Bucket resource.
    :param prefix: When specified, only objects that start with this prefix are
    listed.
    :return: The list of objects.
    """
    try:
        if not prefix:
            objects = list(bucket.objects.all())
        else:
            objects = list(bucket.objects.filter(Prefix=prefix))
            logger.info("Got objects %s from bucket '%s'",
                        [o.key for o in objects], bucket.name)
    except ClientError:
        logger.exception("Couldn't get objects for bucket '%s'.", bucket.name)
        raise
    else:
        return objects
```

- For API details, see [ListObjects in AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end

  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
    count = 0
    puts "The objects in #{@bucket.name} are:"
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
  rescue Aws::Errors::ServiceError => e
```

```
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why: #{e.message}"
    0
end
end

def run_demo
  bucket_name = "doc-example-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [ListObjects in AWS SDK for Ruby API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_objects(client: &Client, bucket_name: &str) -> Result<(), Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;
    println!("Objects in bucket:");
    for obj in objects.contents().unwrap_or_default() {
        println!("{}:{}", obj.key().unwrap());
    }
    Ok(())
}
```

- For API details, see [ListObjects in AWS SDK for Rust API reference](#).

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  oo_result = lo_s3->listobjects(          " oo_result is returned for testing
purpose "
  iv_bucket = iv_bucket_name
```

```
).
MESSAGE 'Retrieved list of object(s) in S3 bucket' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
MESSAGE 'Bucket does not exist' TYPE 'E'.
ENDTRY.
```

- For API details, see [ListObjects in AWS SDK for SAP ABAP API reference](#).

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func listBucketFiles(bucket: String) async throws -> [String] {
    let input = ListObjectsV2Input(
        bucket: bucket
    )
    let output = try await client.listObjectsV2(input: input)
    var names: [String] = []

    guard let objList = output.contents else {
        return []
    }

    for obj in objList {
        if let objName = obj.key {
            names.append(objName)
        }
    }

    return names
}
```

- For API details, see [ListObjects in AWS SDK for Swift API reference](#).

## Restore an archived copy of an object back into an Amazon S3 bucket using an AWS SDK

The following code example shows how to restore an archived copy of an object back into an S3 bucket.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
```

```
try {
    RestoreRequest restoreRequest = RestoreRequest.builder()
        .days(10)

    .glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
        .build();

    RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
        .expectedBucketOwner(expectedBucketOwner)
        .bucket(bucketName)
        .key(keyName)
        .restoreRequest(restoreRequest)
        .build();

    s3.restoreObject(objectRequest);

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [RestoreObject](#) in *AWS SDK for Java 2.x API Reference*.

## Set a new ACL for an Amazon S3 bucket using an AWS SDK

The following code examples show how to set a new access control list (ACL) for an S3 bucket.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void setBucketAcl(S3Client s3, String bucketName, String id) {

    try {
        Grant ownerGrant = Grant.builder()
            .grantee(builder -> builder.id(id))
            .type(Type.CANONICAL_USER)
            .permission(Permission.FULL_CONTROL)
            .build();

        List<Grant> grantList2 = new ArrayList<>();
        grantList2.add(ownerGrant);

        AccessControlPolicy acl = AccessControlPolicy.builder()
            .owner(builder -> builder.id(id))
            .grants(grantList2)
            .build();

        PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
            .bucket(bucketName)
            .accessControlPolicy(acl)
            .build();

        s3.putBucketAcl(putAclReq);
    }
}
```

```
        } catch (S3Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
```

- For API details, see [PutBucketAcl](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Put the bucket ACL.

```
// Import required AWS SDK clients and commands for Node.js.
import { PutBucketAclCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Set the parameters. For more information,
// see https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/
S3.html#putBucketAcl-property.
export const bucketParams = {
    Bucket: "BUCKET_NAME",
    // 'GrantFullControl' allows grantee the read, write, read ACP, and write ACL
    permissions on the bucket.
    // Use a canonical user ID for an AWS account, formatted as follows:
    // id=002160194XXXXXXXXXXXXXXXXXXXXXXa7a49125274
    GrantFullControl: "GRANTEE_1",
    // 'GrantWrite' allows grantee to create, overwrite, and delete any object in the
    bucket.
    // For example, 'uri=http://acs.amazonaws.com/groups/s3/LogDelivery'
    GrantWrite: "GRANTEE_2",
};

export const run = async () => {
    try {
        const data = await s3Client.send(new PutBucketAclCommand(bucketParams));
        console.log("Success, permissions added to bucket", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutBucketAcl](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun setBucketAcl(bucketName: String, idVal: String) {  
  
    val myGrant = Grantee {  
        id = idVal  
        type = Type.CanonicalUser  
    }  
  
    val ownerGrant = Grant {  
        grantee = myGrant  
        permission = Permission.FullControl  
    }  
  
    val grantList = mutableListOf<Grant>()  
    grantList.add(ownerGrant)  
  
    val ownerOb = Owner {  
        id = idVal  
    }  
  
    val acl = AccessControlPolicy {  
        owner = ownerOb  
        grants = grantList  
    }  
  
    val request = PutBucketAclRequest {  
        bucket = bucketName  
        accessControlPolicy = acl  
    }  
  
    S3Client { region = "us-east-1" }.use { s3 ->  
        s3.putBucketAcl(request)  
        println("An ACL was successfully set on $bucketName")  
    }  
}
```

- For API details, see [PutBucketAcl](#) in [AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in  
        Boto3  
                    that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def grant_log_delivery_access(self):  
        """  
        Grant the AWS Log Delivery group write access to the bucket so that  
        Amazon S3 can deliver access logs to the bucket. This is the only  
        recommended  
        use of an S3 bucket ACL.  
        """  
        try:  
            acl = self.bucket.Acl()  
            # Putting an ACL overwrites the existing ACL. If you want to preserve  
            # existing grants, append new grants to the list of existing grants.  
            grants = acl.grants if acl.grants else []  
            grants.append({  
                'Grantee': {  
                    'Type': 'Group',  
                    'URI': 'http://acs.amazonaws.com/groups/s3/LogDelivery'  
                },  
                'Permission': 'WRITE'  
            })  
            acl.put(  
                AccessControlPolicy={  
                    'Grants': grants,  
                    'Owner': acl.owner  
                }  
            )  
            logger.info("Granted log delivery access to bucket '%s'",  
                       self.bucket.name)  
        except ClientError:  
            logger.exception("Couldn't add ACL to bucket '%s'.", self.bucket.name)  
            raise
```

- For API details, see [PutBucketAcl](#) in *AWS SDK for Python (Boto3) API Reference*.

## Set the ACL of an Amazon S3 object using an AWS SDK

The following code example shows how to set the access control list (ACL) of an S3 object.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ObjectWrapper:  
    """Encapsulates S3 object actions."""  
    def __init__(self, s3_object):
```

```
"""
:param s3_object: A Boto3 Object resource. This is a high-level resource in
Boto3
                     that wraps object actions in a class-like structure.
"""
self.object = s3_object
self.key = self.object.key

def put_acl(self, email):
    """
    Applies an ACL to the object that grants read access to an AWS user
identified
    by email address.

    :param email: The email address of the user to grant access.
    """
try:
    acl = self.object.Acl()
    # Putting an ACL overwrites the existing ACL, so append new grants
    # if you want to preserve existing grants.
    grants = acl.grants if acl.grants else []
    grants.append({
        'Grantee': {
            'Type': 'AmazonCustomerByEmail',
            'EmailAddress': email
        },
        'Permission': 'READ'
    })
    acl.put(
        AccessControlPolicy={
            'Grants': grants,
            'Owner': acl.owner
        }
    )
    logger.info("Granted read access to %s.", email)
except ClientError:
    logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
    raise
```

- For API details, see [PutObjectAcl](#) in *AWS SDK for Python (Boto3) API Reference*.

## Set the website configuration for an Amazon S3 bucket using an AWS SDK

The following code examples show how to set the website configuration for an S3 bucket.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::PutWebsiteConfig(const Aws::String &bucketName,
                                   const Aws::String &indexPage, const Aws::String
&errorPage,
                                   const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);
```

```
Aws::S3::Model::IndexDocument indexDocument;
indexDocument.SetSuffix(indexPage);

Aws::S3::Model::ErrorDocument errorDocument;
errorDocument.SetKey(errorPage);

Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
websiteConfiguration.SetIndexDocument(indexDocument);
websiteConfiguration.SetErrorDocument(errorDocument);

Aws::S3::Model::PutBucketWebsiteRequest request;
request.SetBucket(bucketName);
request.SetWebsiteConfiguration(websiteConfiguration);

Aws::S3::Model::PutBucketWebsiteOutcome outcome =
    client.PutBucketWebsite(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: PutBucketWebsite: "
        << outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Success: Set website configuration for bucket '"
        << bucketName << "." << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [PutBucketWebsite](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void setWebsiteConfig( S3Client s3, String bucketName, String
indexDoc) {

    try {
        WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
            .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
            .build();

        PutBucketWebsiteRequest pubWebsiteReq =
PutBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .websiteConfiguration(websiteConfig)
            .build();

        s3.putBucketWebsite(pubWebsiteReq);
        System.out.println("The call was successful");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutBucketWebsite](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Set the website configuration.

```
// Import required AWS SDK clients and commands for Node.js.  
import { PutBucketWebsiteCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an  
Amazon S3 service client module.  
  
// Create the parameters for the bucket  
export const bucketParams = { Bucket: "BUCKET_NAME" };  
export const staticHostParams = {  
    Bucket: bucketParams,  
    WebsiteConfiguration: {  
        ErrorDocument: {  
            Key: "",  
        },  
        IndexDocument: {  
            Suffix: "",  
        },  
    },  
};  
  
export const run = async () => {  
    // Insert specified bucket name and index and error documents into parameters  
    JSON  
    // from command line arguments  
    staticHostParams.Bucket = bucketParams;  
    staticHostParams.WebsiteConfiguration.IndexDocument.Suffix = "INDEX_PAGE"; // The  
index document inserted into parameters JSON.  
    staticHostParams.WebsiteConfiguration.ErrorDocument.Key = "ERROR_PAGE"; // The  
error document inserted into parameters JSON.  
    // Set the new website configuration on the selected bucket.  
    try {  
        const data = await s3Client.send(new  
PutBucketWebsiteCommand(staticHostParams));  
        console.log("Success", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
}
```

```
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutBucketWebsite](#) in [AWS SDK for JavaScript API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"  
  
# Wraps Amazon S3 bucket website actions.  
class BucketWebsiteWrapper  
  attr_reader :bucket_website  
  
  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object  
  # configured with an existing bucket.  
  def initialize(bucket_website)  
    @bucket_website = bucket_website  
  end  
  
  # Sets a bucket as a static website.  
  #  
  # @param index_document [String] The name of the index document for the website.  
  # @param error_document [String] The name of the error document to show for 4XX  
  # errors.  
  # @return [Boolean] True when the bucket is configured as a website; otherwise,  
  # false.  
  def set_website(index_document, error_document)  
    @bucket_website.put(  
      website_configuration: {  
        index_document: { suffix: index_document },  
        error_document: { key: error_document }  
      }  
    )  
    true  
  rescue Aws::Errors::ServiceError => e  
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's  
why: #{e.message}"  
    false  
  end  
end  
  
def run_demo  
  bucket_name = "doc-example-bucket"  
  index_document = "index.html"  
  error_document = "404.html"  
  
  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))  
  return unless wrapper.set_website(index_document, error_document)  
  
  puts "Successfully configured bucket #{bucket_name} as a static website."  
end  
  
run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [PutBucketWebsite](#) in *AWS SDK for Ruby API Reference*.

## Upload a single part of a multipart upload using an AWS SDK

The following code example shows how to upload a single part of a multipart upload.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
let upload_part_res = client
    .upload_part()
    .key(&key)
    .bucket(&bucket_name)
    .upload_id(upload_id)
    .body(stream)
    .part_number(part_number)
    .send()
    .await?;
upload_parts.push(
    CompletedPart::builder()
        .e_tag(upload_part_res.e_tag.unwrap_or_default())
        .part_number(part_number)
        .build(),
);
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();
```

- For API details, see [UploadPart](#) in *AWS SDK for Rust API reference*.

## Upload an object to an Amazon S3 bucket using an AWS SDK

The following code examples show how to upload an object to an S3 bucket.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Shows how to upload a file from the local computer to an Amazon S3
```

```
/// bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The Amazon S3 bucket to which the object
/// will be uploaded.</param>
/// <param name="objectName">The object to upload.</param>
/// <param name="filePath">The path, including file name, of the object
/// on the local computer to upload.</param>
/// <returns>A boolean value indicating the success or failure of the
/// upload procedure.</returns>
public static async Task<bool> UploadFileAsync(
    IAmazonS3 client,
    string bucketName,
    string objectName,
    string filePath)
{
    var request = new PutObjectRequest
    {
        BucketName = bucketName,
        Key = objectName,
        FilePath = filePath,
    };

    var response = await client.PutObjectAsync(request);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"Successfully uploaded {objectName} to
{bucketName}.");
        return true;
    }
    else
    {
        Console.WriteLine($"Could not upload {objectName} to
{bucketName}.");
        return false;
    }
}
```

- For API details, see [PutObject](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::PutObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval
    needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
```

```
Aws::MakeShared< Aws::FStream>("SampleAllocationTag",
                                fileName.c_str(),
                                std::ios_base::in |
                                std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << fileName << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
        s3_client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutObject: " <<
                    outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Added object '" << fileName << "' to bucket '"
                    << bucketName << "'.";
    }

    return outcome.IsSuccess();
}
```

- For API details, see [PutObject](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Place an object in a bucket.
fmt.Println("Upload an object to the bucket")
// Get the object body to upload.
// Image credit: https://unsplash.com/photos/iz58d89q3ss
stat, err := os.Stat("image.jpg")
if err != nil {
    panic("Couldn't stat image: " + err.Error())
}
file, err := os.Open("image.jpg")

if err != nil {
    panic("Couldn't open local file")
}

_, err = client.PutObject(context.TODO(), &s3.PutObjectInput{
    Bucket:      aws.String(name),
    Key:         aws.String("path/myfile.jpg"),
    Body:         file,
    ContentLength: stat.Size(),
})
file.Close()

if err != nil {
    panic("Couldn't upload file: " + err.Error())
}
```

```
}
```

- For API details, see [PutObject](#) in *AWS SDK for Go API Reference*.

Java

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Upload an object to a bucket.

```
public static String putS3Object(S3Client s3, String bucketName, String
objectKey, String objectPath) {

    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("x-amz-meta-myVal", "test");
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .metadata(metadata)
            .build();

        PutObjectResponse response = s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));
        return response.eTag();

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    return "";
}

// Return a byte array.
private static byte[] getObjectFile(String filePath) {

    FileInputStream fileInputStream = null;
    byte[] bytesArray = null;

    try {
        File file = new File(filePath);
        bytesArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(bytesArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```
        return bytesArray;
    }
```

Upload an object to a bucket and set tags.

```
public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {

    try {

        Tag tag1 = Tag.builder()
            .key("Tag 1")
            .value("This is tag 1")
            .build();

        Tag tag2 = Tag.builder()
            .key("Tag 2")
            .value("This is tag 2")
            .build();

        List<Tag> tags = new ArrayList<>();
        tags.add(tag1);
        tags.add(tag2);

        Tagging allTags = Tagging.builder()
            .tagSet(tags)
            .build();

        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .tagging(allTags)
            .build();

        s3.putObject(putOb, RequestBody.fromBytes(getObjectFile(objectPath)));

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {

    try {
        GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
        List<Tag> obTags = getTaggingRes.tagSet();
        for (Tag sinTag: obTags) {
            System.out.println("The tag key is: "+sinTag.key());
            System.out.println("The tag value is: "+sinTag.value());
        }

        // Replace the object's tags with two new tags.
        Tag tag3 = Tag.builder()
            .key("Tag 3")
    }
}
```

```

        .value("This is tag 3")
        .build();

    Tag tag4 = Tag.builder()
        .key("Tag 4")
        .value("This is tag 4")
        .build();

    List<Tag> tags = new ArrayList<>();
    tags.add(tag3);
    tags.add(tag4);

    Tagging updatedTags = Tagging.builder()
        .tagSet(tags)
        .build();

    PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .tagging(updatedTags)
        .build();

    s3.putObjectTagging(taggingRequest1);
    GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
    List<Tag> modTags = getTaggingRes2.tagSet();
    for (Tag sinTag: modTags) {
        System.out.println("The tag key is: "+sinTag.key());
        System.out.println("The tag value is: "+sinTag.value());
    }

} catch (S3Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

```

```
Upload an object to a bucket and set metadata.

public static String putS3Object(S3Client s3, String bucketName, String
objectKey, String objectPath) {

    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("author", "Mary Doe");
        metadata.put("version", "1.0.0.0");

        PutObjectRequest put0b = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .metadata(metadata)
            .build();

        PutObjectResponse response = s3.putObject(put0b,
RequestBody.fromBytes(getObjectFile(objectPath)));
        return response.eTag();

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    return "";
}
```

```
}

// Return a byte array.
private static byte[] getObjectFile(String filePath) {

    FileInputStream fileInputStream = null;
    byte[] byteArray = null;

    try {
        File file = new File(filePath);
        byteArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(byteArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

return byteArray;
}
```

Upload an object to a bucket and set an object retention value.

```
public static void setRetentionPeriod(S3Client s3, String key, String bucket) {

    try{
        LocalDate localDate = LocalDate.parse("2020-07-17");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        ObjectLockRetention lockRetention = ObjectLockRetention.builder()
            .mode("COMPLIANCE")
            .retainUntilDate(instant)
            .build();

        PutObjectRetentionRequest retentionRequest =
        PutObjectRetentionRequest.builder()
            .bucket(bucket)
            .key(key)
            .bypassGovernanceRetention(true)
            .retention(lockRetention)
            .build();

        // To set Retention on an object, the Amazon S3 bucket must support
        // object locking, otherwise an exception is thrown.
        s3.putObjectRetention(retentionRequest);
        System.out.print("An object retention configuration was successfully
placed on the object");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutObject](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Create and upload the object.

```
// Import required AWS SDK clients and commands for Node.js.  
import { PutObjectCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an  
Amazon S3 service client module.  
  
// Set the parameters.  
export const bucketParams = {  
    Bucket: "BUCKET_NAME",  
    // Specify the name of the new object. For example, 'index.html'.  
    // To create a directory for the object, use '/'. For example, 'myApp/  
    package.json'.  
    Key: "OBJECT_NAME",  
    // Content of the new object.  
    Body: "BODY",  
};  
  
// Create and upload the object to the S3 bucket.  
export const run = async () => {  
    try {  
        const data = await s3Client.send(new PutObjectCommand(bucketParams));  
        return data; // For unit tests.  
        console.log(  
            "Successfully uploaded object: " +  
            bucketParams.Bucket +  
            "/" +  
            bucketParams.Key  
        );  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

Upload the object.

```
// Import required AWS SDK clients and commands for Node.js.
```

```
import { PutObjectCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
// Amazon S3 service client module.
import {path} from "path";
import {fs} from "fs";

const file = "OBJECT_PATH_AND_NAME"; // Path to and name of object. For example
'./myFiles/index.js'.
const fileStream = fs.createReadStream(file);

// Set the parameters
export const uploadParams = {
  Bucket: "BUCKET_NAME",
  // Add the required 'Key' parameter using the 'path' module.
  Key: path.basename(file),
  // Add the required 'Body' parameter
  Body: fileStream,
};

// Upload file to specified bucket.
export const run = async () => {
  try {
    const data = await s3Client.send(new PutObjectCommand(uploadParams));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutObject](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun putS3Object(bucketName: String, objectKey: String, objectPath: String)
{
  val metadataVal = mutableMapOf<String, String>()
  metadataVal["myVal"] = "test"

  val request = PutObjectRequest {
    bucket = bucketName
    key = objectKey
    metadata = metadataVal
    body = File(objectPath).asByteStream()
  }
```

```
S3Client { region = "us-east-1" }.use { s3 ->
    val response = s3.putObject(request)
    println("Tag information is ${response.eTag}")
}
}
```

- For API details, see [PutObject](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Upload an object to a bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);

$file_name = "local-file-" . uniqid();
try {
    $s3client->putObject([
        'Bucket' => $bucket_name,
        'Key' => $file_name,
        'SourceFile' => 'testfile.txt'
    ]);
    echo "Uploaded $file_name to $bucket_name.\n";
} catch (Exception $exception) {
    echo "Failed to upload $file_name with error: " . $exception->getMessage();
    exit("Please fix error with file upload before continuing.");
}
```

- For API details, see [PutObject](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
        Boto3
                    that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def put(self, data):
        """
        Upload data to the object.
        """

```

```
    :param data: The data to upload. This can either be bytes or a string. When
this
                    argument is a string, it is interpreted as a file name, which
is
                    opened in read bytes mode.
    """
    put_data = data
    if isinstance(data, str):
        try:
            put_data = open(data, 'rb')
        except IOError:
            logger.exception("Expected file name or binary data, got '%s'.",
data)
            raise

        try:
            self.object.put(Body=put_data)
            self.object.wait_until_exists()
            logger.info(
                "Put object '%s' to bucket '%s'.", self.object.key,
                self.object.bucket_name)
        except ClientError:
            logger.exception(
                "Couldn't put object '%s' to bucket '%s'.", self.object.key,
                self.object.bucket_name)
            raise
    finally:
        if getattr(put_data, 'close', None):
            put_data.close()
```

- For API details, see [PutObject](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Upload a file using a managed uploader (`Object.upload_file`).

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
#{e.message}"
```

```

        false
    end
end

def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
  object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

Upload a file using Object.put.

```

require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, "rb") do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
  #{e.message}"
    false
  end
end

def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

Upload a file using Object.put and add server-side encryption.

```

require "aws-sdk-s3"

```

```
# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
  object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
  #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [PutObject](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn upload_object(
  client: &Client,
  bucket_name: &str,
  file_name: &str,
  key: &str,
) -> Result<(), Error> {
  let body = ByteStream::from_path(Path::new(file_name)).await;
  client
    .put_object()
    .bucket(bucket_name)
    .key(key)
    .body(body.unwrap())
    .send()
```

```
    .await?;

    println!("Uploaded file: {}", file_name);
    Ok(())
}
```

- For API details, see [PutObject](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Get contents of file from application server"
DATA lv_body TYPE xstring.
OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.
READ DATASET iv_file_name INTO lv_body.
CLOSE DATASET iv_file_name.

"Upload/Put an object to S3 bucket"
TRY.
    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_file_name
        iv_body = lv_body
    ).
    MESSAGE 'Object uploaded to S3 bucket' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist' TYPE 'E'.
ENDTRY.
```

- For API details, see [PutObject](#) in *AWS SDK for SAP ABAP API reference*.

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Upload a file from local storage to a bucket.

```
public func uploadFile(bucket: String, key: String, file: String) async throws
{
```

```
let fileUrl = URL(fileURLWithPath: file)
let fileData = try Data(contentsOf: fileUrl)
let dataStream = ByteStream.from(data: fileData)

let input = PutObjectInput(
    body: dataStream,
    bucket: bucket,
    key: key
)
_ = try await client.putObject(input: input)
}
```

Upload the contents of a Swift Data object to a bucket.

```
public func createFile(bucket: String, key: String, withData data: Data) async
throws {
    let dataStream = ByteStream.from(data: data)

    let input = PutObjectInput(
        body: dataStream,
        bucket: bucket,
        key: key
    )
    _ = try await client.putObject(input: input)
}
```

- For API details, see [PutObject](#) in *AWS SDK for Swift API reference*.

## Scenarios for Amazon S3 using AWS SDKs

The following code examples show how to use Amazon Simple Storage Service with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Create a presigned URL for Amazon S3 using an AWS SDK \(p. 1757\)](#)
- [Get started with Amazon S3 buckets and objects using an AWS SDK \(p. 1765\)](#)
- [Manage versioned Amazon S3 objects in batches with a Lambda function using an AWS SDK \(p. 1802\)](#)
- [Upload or download large files to and from Amazon S3 using an AWS SDK \(p. 1803\)](#)
- [Work with Amazon S3 versioned objects using an AWS SDK \(p. 1820\)](#)

## Create a presigned URL for Amazon S3 using an AWS SDK

The following code examples show how to create a presigned URL for S3 and upload an object.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get a presigned URL for the object.  
// In order to get a presigned URL for an object, you must  
// create a Presignclient  
fmt.Println("Create Presign client")  
presignClient := s3.NewPresignClient(&client)  
  
presignParams := &s3.GetObjectInput{  
    Bucket: aws.String(name),  
    Key: aws.String("path/myfile.jpg"),  
}  
  
// Apply an expiration via an option function  
presignDuration := func(po *s3.PresignOptions) {  
    po.Expires = 5 * time.Minute  
}  
  
presignResult, err := presignClient.PresignGetObject(context.TODO(),  
    presignParams, presignDuration)  
  
if err != nil {  
    panic("Couldn't get presigned URL for GetObject")  
}  
  
fmt.Printf("Presigned URL For object: %s\n", presignResult.URL)
```

Java

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void signBucket(S3Presigner presigner, String bucketName, String  
keyName) {  
  
    try {  
        PutObjectRequest objectRequest = PutObjectRequest.builder()  
            .bucket(bucketName)  
            .key(keyName)  
            .contentType("text/plain")  
            .build();  
  
        PutObjectPresignRequest presignRequest =  
PutObjectPresignRequest.builder()  
            .signatureDuration(Duration.ofMinutes(10))  
            .putObjectRequest(objectRequest)  
            .build();  
  
        PresignedPutObjectRequest presignedRequest =  
presigner.presignPutObject(presignRequest);  
        String myURL = presignedRequest.url().toString();  
        System.out.println("Presigned URL to upload a file to: " +myURL);  
        System.out.println("Which HTTP method needs to be used when uploading a  
file: " + presignedRequest.httpRequest().method());  
  
        // Upload content to the Amazon S3 bucket by using this URL.  
        URL url = presignedRequest.url();
```

```
// Create the connection and use it to upload the new object by using
// the presigned URL.
HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
connection.setDoOutput(true);
connection.setRequestProperty("Content-Type","text/plain");
connection.setRequestMethod("PUT");
OutputStreamWriter out = new
OutputStreamWriter(connection.getOutputStream());
out.write("This text was uploaded as an object by using a presigned
URL.");
out.close();

connection.getResponseCode();
System.out.println("HTTP response code is " +
connection.getResponseCode());

} catch (S3Exception | IOException e) {
    e.printStackTrace();
}
}
```

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Create a presigned URL to upload an object to a bucket.

```
// Import the required AWS SDK clients and commands for Node.js
import {
    CreateBucketCommand,
    DeleteObjectCommand,
    PutObjectCommand,
    DeleteBucketCommand
} from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
// Amazon S3 service client module.
import { getSignedUrl } from "@aws-sdk/s3-request-presigner";
import fetch from "node-fetch";

// Set parameters
// Create a random name for the Amazon Simple Storage Service (Amazon S3) bucket
// and key
export const bucketParams = {
    Bucket: `test-bucket-${Math.ceil(Math.random() * 10 ** 10)}`,
    Key: `test-object-${Math.ceil(Math.random() * 10 ** 10)}`,
    Body: "BODY"
```

```

};

export const run = async () => {
  try {
    // Create an S3 bucket.
    console.log(`Creating bucket ${bucketParams.Bucket}`);
    await s3Client.send(new CreateBucketCommand({ Bucket: bucketParams.Bucket }));
    console.log(`Waiting for "${bucketParams.Bucket}" bucket creation...`);
  } catch (err) {
    console.log("Error creating bucket", err);
  }
  try {
    // Create a command to put the object in the S3 bucket.
    const command = new PutObjectCommand(bucketParams);
    // Create the presigned URL.
    const signedUrl = await getSignedUrl(s3Client, command, {
      expiresIn: 3600,
    });
    console.log(
      `\nPutting "${bucketParams.Key}" using signedUrl with body
      "${bucketParams.Body}" in v3`
    );
    console.log(signedUrl);
    const response = await fetch(signedUrl, {method: 'PUT', body:
      bucketParams.Body});
    console.log(
      `\nResponse returned by signed URL: ${await response.text()}\n`
    );
  } catch (err) {
    console.log("Error creating presigned URL", err);
  }
  try {
    // Delete the object.
    console.log(`\nDeleting object "${bucketParams.Key}" from bucket`);
    await s3Client.send(
      new DeleteObjectCommand({ Bucket: bucketParams.Bucket, Key:
        bucketParams.Key })
    );
  } catch (err) {
    console.log("Error deleting object", err);
  }
  try {
    // Delete the S3 bucket.
    console.log(`\nDeleting bucket ${bucketParams.Bucket}`);
    await s3Client.send(
      new DeleteBucketCommand({ Bucket: bucketParams.Bucket })
    );
  } catch (err) {
    console.log("Error deleting bucket", err);
  }
};
run();

```

Create a presigned URL to download an object from a bucket.

```

// Import the required AWS SDK clients and commands for Node.js
import {
  CreateBucketCommand,
  PutObjectCommand,
  GetObjectCommand,
  DeleteObjectCommand,
  DeleteBucketCommand
} from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

```

```
import { getSignedUrl } from "@aws-sdk/s3-request-presigner";
const fetch = require("node-fetch");

// Set parameters
// Create a random names for the S3 bucket and key.
export const bucketParams = {
  Bucket: `test-bucket-${Math.ceil(Math.random() * 10 ** 10)}`,
  Key: `test-object-${Math.ceil(Math.random() * 10 ** 10)}`,
  Body: "BODY"
};

export const run = async () => {
  // Create an S3 bucket.
  try {
    console.log(`Creating bucket ${bucketParams.Bucket}`);
    const data = await s3Client.send(
      new CreateBucketCommand({ Bucket: bucketParams.Bucket })
    );
    return data; // For unit tests.
    console.log(`Waiting for "${bucketParams.Bucket}" bucket creation...\n`);
  } catch (err) {
    console.log("Error creating bucket", err);
  }
  // Put the object in the S3 bucket.
  try {
    console.log(`Putting object "${bucketParams.Key}" in bucket`);
    const data = await s3Client.send(
      new PutObjectCommand({
        Bucket: bucketParams.Bucket,
        Key: bucketParams.Key,
        Body: bucketParams.Body,
      })
    );
    return data; // For unit tests.
  } catch (err) {
    console.log("Error putting object", err);
  }
  // Create a presigned URL.
  try {
    // Create the command.
    const command = new GetObjectCommand(bucketParams);

    // Create the presigned URL.
    const signedUrl = await getSignedUrl(s3Client, command, {
      expiresIn: 3600,
    });
    console.log(
      `\nGetting "${bucketParams.Key}" using signedUrl with body
      "${bucketParams.Body}" in v3`
    );
    console.log(signedUrl);
    const response = await fetch(signedUrl);
    console.log(
      `\nResponse returned by signed URL: ${await response.text()}\n`
    );
  } catch (err) {
    console.log("Error creating presigned URL", err);
  }
  // Delete the object.
  try {
    console.log(`\nDeleting object "${bucketParams.Key}" from bucket`);
    const data = await s3Client.send(
      new DeleteObjectCommand({ Bucket: bucketParams.Bucket, Key: bucketParams.Key })
    );
    return data; // For unit tests.
  }
}
```

```
        } catch (err) {
            console.log("Error deleting object", err);
        }
        // Delete the S3 bucket.
        try {
            console.log(`\nDeleting bucket ${bucketParams.Bucket}`);
            const data = await s3Client.send(
                new DeleteBucketCommand({ Bucket: bucketParams.Bucket, Key: bucketParams.Key })
            );
            return data; // For unit tests.
        } catch (err) {
            console.log("Error deleting object", err);
        }
    };
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Generate a presigned URL that can perform an S3 action for a limited time. Use the Requests package to make a request with the URL.

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)

def generate_presigned_url(s3_client, client_method, method_parameters,
                           expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method,
            Params=method_parameters,
            ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.", client_method)
        raise
    return url
```

```

def usage_demo():
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print('*'*88)

    parser = argparse.ArgumentParser()
    parser.add_argument('bucket', help="The name of the bucket.")
    parser.add_argument(
        'key', help="For a GET operation, the key of the object in Amazon S3. For a
"
        "PUT operation, the name of a file to upload.")
    parser.add_argument(
        'action', choices=('get', 'put'), help="The action to perform.")
    args = parser.parse_args()

    s3_client = boto3.client('s3')
    client_action = 'get_object' if args.action == 'get' else 'put_object'
    url = generate_presigned_url(
        s3_client, client_action, {'Bucket': args.bucket, 'Key': args.key}, 1000)

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == 'get':
        response = requests.get(url)
    elif args.action == 'put':
        print("Putting data to the URL.")
        try:
            with open(args.key, 'r') as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(f"Couldn't find {args.key}. For a PUT operation, the key must be
the "
                  f"name of a file that exists on your computer.")

    if response is not None:
        print("Got response:")
        print(f"Status: {response.status_code}")
        print(response.text)

    print('*'*88)

if __name__ == '__main__':
    usage_demo()

```

Generate a presigned POST request to upload a file.

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                           that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def generate_presigned_post(self, object_key, expires_in):
        """
        """

```

AWS

```
Generate a presigned Amazon S3 POST request to upload a file.  
A presigned POST can be used for a limited time to let someone without an  
account upload a file to a bucket.  
  
:param object_key: The object key to identify the uploaded object.  
:param expires_in: The number of seconds the presigned POST is valid.  
:return: A dictionary that contains the URL and form fields that contain  
required access data.  
"""  
try:  
    response = self.bucket.meta.client.generate_presigned_post(  
        Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in)  
    logger.info("Got presigned POST URL: %s", response['url'])  
except ClientError:  
    logger.exception(  
        "Couldn't get a presigned POST URL for bucket '%s' and object  
'%s'",  
        self.bucket.name, object_key)  
    raise  
return response
```

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"  
require "net/http"  
  
# Creates a presigned URL that can be used to upload content to an object.  
#  
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.  
# @param object_key [String] The key to give the uploaded object.  
# @return [URI, nil] The parsed URI if successful; otherwise nil.  
def get_presigned_url(bucket, object_key)  
    url = bucket.object(object_key).presigned_url(:put)  
    puts "Created presigned URL: #{url}."  
    URI(url)  
rescue Aws::Errors::ServiceError => e  
    puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's why:  
#{e.message}"  
end  
  
def run_demo  
    bucket_name = "doc-example-bucket"  
    object_key = "my-file.txt"  
    object_content = "This is the content of my-file.txt."  
  
    bucket = Aws::S3::Bucket.new(bucket_name)  
    presigned_url = get_presigned_url(bucket, object_key)  
    return unless presigned_url  
  
    response = Net::HTTP.start(presigned_url.host) do |http|  
        http.send_request("PUT", presigned_url.request_uri, object_content,  
        "content_type" => "")  
    end  
  
    case response
```

```
when Net::HTTPSuccess
  puts "Content uploaded!"
else
  puts response.value
end
end

run_demo if $PROGRAM_NAME == __FILE__
```

## Get started with Amazon S3 buckets and objects using an AWS SDK

The following code examples show how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.
- Delete a bucket.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;

public class S3_Basics
{
    public static async Task Main()
    {
        // Create an Amazon S3 client object. The constructor uses the
        // default user installed on the system. To work with Amazon S3
        // features in a different AWS Region, pass the AWS Region as a
        // parameter to the client constructor.
        IAmazonS3 client = new AmazonS3Client();
        string bucketName = string.Empty;
        string filePath = string.Empty;
        string keyName = string.Empty;

        Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
        Console.WriteLine("procedures. This application will:");
        Console.WriteLine("\n\t1. Create a bucket");
        Console.WriteLine("\n\t2. Upload an object to the new bucket");
        Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
bucket");
        Console.WriteLine("\n\t4. List the items in the new bucket");
        Console.WriteLine("\n\t5. Delete all the items in the bucket");
```

```
Console.WriteLine("\n\t6. Delete the bucket");

Console.WriteLine("-----");

// Create a bucket.
Console.WriteLine("\nCreate a new Amazon S3 bucket.\n");

Console.Write("Please enter a name for the new bucket: ");
bucketName = Console.ReadLine();

var success = await S3Bucket.CreateBucketAsync(client, bucketName);
if (success)
{
    Console.WriteLine($"Successfully created bucket: {bucketName}.\n");
}
else
{
    Console.WriteLine($"Could not create bucket: {bucketName}.\n");
}

Console.WriteLine("Upload a file to the new bucket.");

// Get the local path and filename for the file to upload.
while (string.IsNullOrEmpty(filePath))
{
    Console.Write("Please enter the path and filename of the file to
upload: ");
    filePath = Console.ReadLine();

    // Confirm that the file exists on the local computer.
    if (!File.Exists(filePath))
    {
        Console.WriteLine($"Couldn't find {filePath}. Try again.\n");
        filePath = string.Empty;
    }
}

// Get the file name from the full path.
keyName = Path.GetFileName(filePath);

success = await S3Bucket.UploadFileAsync(client, bucketName, keyName,
filePath);

if (success)
{
    Console.WriteLine($"Successfully uploaded {keyName} from {filePath}
to {bucketName}.\n");
}
else
{
    Console.WriteLine($"Could not upload {keyName}.\n");
}

// Set the file path to an empty string to avoid overwriting the
// file we just uploaded to the bucket.
filePath = string.Empty;

// Now get a new location where we can save the file.
while (string.IsNullOrEmpty(filePath))
{
    // First get the path to which the file will be downloaded.
    Console.Write("Please enter the path where the file will be
downloaded: ");
    filePath = Console.ReadLine();

    // Confirm that the file exists on the local computer.
```

```

        if (File.Exists($"{filePath}\\{keyName}"))
        {
            Console.WriteLine($"Sorry, the file already exists in that
location.\n");
            filePath = string.Empty;
        }
    }

    // Download an object from a bucket.
    success = await S3Bucket.DownloadObjectFromBucketAsync(client,
bucketName, keyName, filePath);

    if (success)
    {
        Console.WriteLine($"Successfully downloaded {keyName}.\n");
    }
    else
    {
        Console.WriteLine($"Sorry, could not download {keyName}.\n");
    }

    // Copy the object to a different folder in the bucket.
    string folderName = string.Empty;

    while (string.IsNullOrEmpty(folderName))
    {
        Console.Write("Please enter the name of the folder to copy your
object to: ");
        folderName = Console.ReadLine();
    }

    while (string.IsNullOrEmpty(keyName))
    {
        // Get the name to give to the object once uploaded.
        Console.Write("Enter the name of the object to copy: ");
        keyName = Console.ReadLine();
    }

    await S3Bucket.CopyObjectInBucketAsync(client, bucketName, keyName,
folderName);

    // List the objects in the bucket.
    await S3Bucket.ListBucketContentsAsync(client, bucketName);

    // Delete the contents of the bucket.
    await S3Bucket.DeleteBucketContentsAsync(client, bucketName);

    // Deleting the bucket too quickly after deleting its contents will
    // cause an error that the bucket isn't empty. So...
    Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
    _ = Console.ReadLine();

    // Delete the bucket.
    await S3Bucket.DeleteBucketAsync(client, bucketName);
}
}

```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)

- [DeleteObjects](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsRequest.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <aws/core/utils/memory/stl/AWSStreamFwd.h>
#include <fstream>
#include "awsdoc/s3/s3_examples.h"

namespace AwsDoc {
    namespace S3 {

        ///! Delete an S3 bucket.
        /*!
         \sa DeleteBucket()
         \param bucketName The S3 bucket's name.
         \param client An S3 client.
        */
        static bool DeleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

        ///! Delete an object in an S3 bucket.
        /*!
         \sa DeleteObjectFromBucket()
         \param bucketName The S3 bucket's name.
         \param key The key for the object in the S3 bucket.
         \param client An S3 client.
        */
        static bool
        DeleteObjectFromBucket(const Aws::String &bucketName, const Aws::String &key, Aws::S3::S3Client &client);
    }
}

///! Scenario to create, copy, and delete S3 buckets and objects.
/*!
 \sa S3_GettingStartedScenario()
 \param uploadFilePath Path to file to upload to an Amazon S3 bucket.
 \param saveFilePath Path for saving a downloaded S3 object.
```

```
\param clientConfig Aws client configuration.  
*/  
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &uploadFilePath, const  
Aws::String &saveFilePath,  
                                              const Aws::Client::ClientConfiguration  
&clientConfig) {  
  
    Aws::S3::S3Client client(clientConfig);  
  
    // Create a unique bucket name which is only temporary and will be deleted.  
    // Format: "doc-example-bucket-" + lowercase UUID.  
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();  
    Aws::String bucketName = "doc-example-bucket-" +  
                           Aws::Utils::StringUtils::ToLower(uuid.c_str());  
  
    // 1. Create a bucket.  
    {  
        Aws::S3::Model::CreateBucketRequest request;  
        request.SetBucket(bucketName);  
  
        if (clientConfig.region != Aws::Region::US_EAST_1) {  
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;  
            createBucketConfiguration.WithLocationConstraint(  
Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(  
                                         clientConfig.region));  
            request.WithCreateBucketConfiguration(createBucketConfiguration);  
        }  
  
        Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);  
  
        if (!outcome.IsSuccess()) {  
            const Aws::S3::S3Error &err = outcome.GetError();  
            std::cerr << "Error: CreateBucket: " <<  
                  err.GetExceptionName() << ":" << err.GetMessage() <<  
std::endl;  
            return false;  
        }  
        else {  
            std::cout << "Created the bucket, '" << bucketName <<  
                  "', in the region, '" << clientConfig.region << "." <<  
std::endl;  
        }  
    }  
  
    // 2. Upload a local file to the bucket.  
    Aws::String key = "key-for-test";  
    {  
        Aws::S3::Model::PutObjectRequest request;  
        request.SetBucket(bucketName);  
        request.SetKey(key);  
  
        std::shared_ptr<Aws::FStream> input_data =  
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",  
                                         uploadFilePath,  
                                         std::ios_base::in |  
                                         std::ios_base::binary);  
  
        if (!input_data->is_open()) {  
            std::cerr << "Error: unable to open file, '" << uploadFilePath << "'."  
<< std::endl;  
            AwsDoc::S3::DeleteBucket(bucketName, client);  
            return false;  
        }  
  
        request.SetBody(input_data);  
    }
```

```

Aws::S3::Model::PutObjectOutcome outcome =
    client.PutObject(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: PutObject: " <<
        outcome.GetError().GetMessage() << std::endl;
    AwsDoc::S3::DeleteObjectFromBucket(bucketName, key, client);
    AwsDoc::S3::DeleteBucket(bucketName, client);
    return false;
}
else {
    std::cout << "Added the object with the key, '" << key << "', to the
bucket, '''
                << bucketName << "." << std::endl;
}
}

// 3. Download the object to a local file.
{
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: GetObject: " <<
            err.GetExceptionName() << ":" << err.GetMessage() <<
        std::endl;
    }
    else {
        std::cout << "Downloaded the object with the key, '" << key << "', in
the bucket, '''
                << bucketName << "." << std::endl;

        Aws::Iostream &ioStream = outcome.GetResultWithOwnership().
            GetBody();
        Aws::ofstream outStream(saveFilePath);
        if (!outStream.is_open()) {
            std::cout << "Error: unable to open file, '" << saveFilePath <<
".\" << std::endl;
        }
        else {
            outStream << ioStream.rdbuf();
            std::cout << "Wrote the downloaded object to the file ''"
                << saveFilePath << "." << std::endl;
        }
    }
}

// 4. Copy the object to a different "folder" in the bucket.
Aws::String copiedToKey = "test-folder/" + key;
{
    Aws::S3::Model::CopyObjectRequest request;
    request.WithBucket(bucketName)
        .WithKey(copiedToKey)
        .WithCopySource(bucketName + "/" + key);

    Aws::S3::Model::CopyObjectOutcome outcome =
        client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error: CopyObject: " <<
            outcome.GetError().GetMessage() << std::endl;
    }
}

```

```

        }
        else {
            std::cout << "Copied the object with the key, '" << key << "', to the
key, '" << copiedToKey
                           << ", in the bucket, '" << bucketName << "." << std::endl;
        }
    }

// 5. List objects in the bucket.
{
    Aws::S3::Model::ListObjectsRequest request;
    request.WithBucket(bucketName);

    Aws::S3::Model::ListObjectsOutcome outcome = client.ListObjects(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ListObjects: " <<
                    outcome.GetError().GetMessage() << std::endl;
    }
    else {
        Aws::Vector<Aws::S3::Model::Object> objects =
            outcome.GetResult().GetContents();

        std::cout << objects.size() << " objects in the bucket, '" <<
bucketName << ":" << std::endl;

        for (Aws::S3::Model::Object &object: objects) {
            std::cout << "      '" << object.GetKey() << "' " << std::endl;
        }
    }
}

// 6. Delete all objects in the bucket.
// All objects in the bucket must be deleted before deleting the bucket.
AwsDoc::S3::DeleteObjectFromBucket(bucketName, copiedToKey, client);
AwsDoc::S3::DeleteObjectFromBucket(bucketName, key, client);

// 7. Delete the bucket.
return AwsDoc::S3::DeleteBucket(bucketName, client);
}

bool AwsDoc::S3::DeleteObjectFromBucket(const Aws::String &bucketName, const
                                         Aws::String &key,
                                         Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: DeleteObject: " <<
                    outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Deleted the object with the key, '" << key << "', from the
bucket, '" << bucketName << "." << std::endl;
    }
}

return outcome.IsSuccess();
}

bool AwsDoc::S3::DeleteBucket(const Aws::String &bucketName, Aws::S3::S3Client
&client) {

```

```
Aws::S3::Model::DeleteBucketRequest request;
request.SetBucket(bucketName);

Aws::S3::Model::DeleteBucketOutcome outcome =
    client.DeleteBucket(request);

if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &err = outcome.GetError();
    std::cerr << "Error: DeleteBucket: " <<
        err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
}
else {
    std::cout << "Deleted the bucket, '" << bucketName << "'." << std::endl;
}
return outcome.IsSuccess();
}
```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// This bucket name is 100% unique.
// Remember that bucket names must be globally unique among all buckets.

myBucketName := "mybucket-" + (xid.New().String())
fmt.Printf("Bucket name: %v\n", myBucketName)

cfg, err := config.LoadDefaultConfig(context.TODO())

if err != nil {
    panic("Failed to load configuration")
}

s3client := s3.NewFromConfig(cfg)

MakeBucket(*s3client, myBucketName)
Bucket0ps(*s3client, myBucketName)
AccountBucketOps(*s3client, myBucketName)
BucketDel0ps(*s3client, myBucketName)
```

- For API details, see the following topics in *AWS SDK for Go API Reference*.
  - [CopyObject](#)

- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
 * Before running this Java V2 code example, set up your development environment,  
 * including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * This Java code example performs the following tasks:  
 *  
 * 1. Creates an Amazon S3 bucket.  
 * 2. Uploads an object to the bucket.  
 * 3. Downloads the object to another local file.  
 * 4. Uploads an object using multipart upload.  
 * 5. List all objects located in the Amazon S3 bucket.  
 * 6. Copies the object to another Amazon S3 bucket.  
 * 7. Deletes the object from the Amazon S3 bucket.  
 * 8. Deletes the Amazon S3 bucket.  
 */  
  
public class S3Scenario {  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    public static void main(String[] args) throws IOException {  
  
        final String usage = "\n" +  
            "Usage:\n" +  
            "  <bucketName> <key> <objectPath> <savePath> <toBucket>\n\n" +  
            "Where:\n" +  
            "  <bucketName> - The Amazon S3 bucket to create.\n\n" +  
            "  <key> - The key to use.\n\n" +  
            "  <objectPath> - The path where the file is located (for example, C:/  
AWS/book2.pdf). "+  
            "  <savePath> - The path where the file is saved after it's downloaded  
        (for example, C:/AWS/book2.pdf). " +  
            "  <toBucket> - An Amazon S3 bucket to where an object is copied to  
        (for example, C:/AWS/book2.pdf). ";  
  
        if (args.length != 5) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String bucketName = args[0];  
        String key = args[1];
```

```
String objectPath = args[2];
String savePath = args[3];
String toBucket = args[4] ;

ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .credentialsProvider(credentialsProvider)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon S3 example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an Amazon S3 bucket.");
createBucket(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Update a local file to the Amazon S3 bucket.");
uploadLocalFile(s3, bucketName, key, objectPath);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Download the object to another local file.");
getObjectBytes(s3, bucketName, key, savePath);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Perform a multipart upload.");
String multipartKey = "multiPartKey";
multipartUpload(s3, toBucket, multipartKey);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. List all objects located in the Amazon S3 bucket.");
listAllObjects(s3, bucketName);
anotherListExample(s3, bucketName) ;
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Copy the object to another Amazon S3 bucket.");
copyBucketObject(s3, bucketName, key, toBucket);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the object from the Amazon S3 bucket.");
deleteObjectFromBucket(s3, bucketName, key);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Delete the Amazon S3 bucket.");
deleteBucket(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("All Amazon S3 operations were successfully performed");
System.out.println(DASHES);
s3.close();
}

// Create a bucket by using a S3Waiter object
public static void createBucket( S3Client s3Client, String bucketName) {
```

```

try {
    S3Waiter s3Waiter = s3Client.waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    s3Client.createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    WaiterResponse<HeadBucketResponse> waiterResponse =
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println(bucketName + " is ready");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void deleteBucket(S3Client client, String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    client.deleteBucket(deleteBucketRequest);
    System.out.println(bucket + " was deleted.");
}

/**
 * Upload an object in parts
 */
private static void multipartUpload(S3Client s3, String bucketName, String key)
{
    int mB = 1024 * 1024;
    // First create a multipart upload and get the upload id
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();

    CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
    String uploadId = response.uploadId();
    System.out.println(uploadId);

    // Upload all the different parts of the object
    UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .partNumber(1).build();

    String etag1 = s3.uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB))).eTag();
    CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

    UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
        .uploadId(uploadId)
        .partNumber(2).build();
}

```

```
        String etag2 = s3.uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB))).eTag();
        CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

        // Call completeMultipartUpload operation to tell S3 to merge all uploaded
        // parts and finish the multipart operation.
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
        .parts(part1, part2)
        .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .multipartUpload(completedMultipartUpload)
        .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
    }

private static ByteBuffer getRandomByteBuffer(int size) {
    byte[] b = new byte[size];
    new Random().nextBytes(b);
    return ByteBuffer.wrap(b);
}

// Return a byte array
private static byte[] getObjectFile(String filePath) {
    FileInputStream fileInputStream = null;
    byte[] bytesArray = null;

    try {
        File file = new File(filePath);
        bytesArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(bytesArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    return bytesArray;
}

public static void getObjectBytes (S3Client s3, String bucketName, String
keyName, String path ) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
        .builder()
        .key(keyName)
        .bucket(bucketName)
        .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
        byte[] data = objectBytes.asByteArray();
    }
}
```

```
// Write the data to a local file.
File myFile = new File(path);
OutputStream os = new FileOutputStream(myFile);
os.write(data);
System.out.println("Successfully obtained bytes from an S3 object");
os.close();

} catch (IOException ex) {
    ex.printStackTrace();
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void uploadLocalFile(S3Client s3, String bucketName, String key,
String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    s3.putObject(objectRequest,
    RequestBody.fromBytes(getObjectFile(objectPath)));
}

public static void listAllObjects(S3Client s3, String bucketName) {

    ListObjectsV2Request listObjectsReqManual = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    boolean done = false;
    while (!done) {
        ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
        for (S3Object content : listObjResponse.contents()) {
            System.out.println(content.key());
        }

        if (listObjResponse.nextContinuationToken() == null) {
            done = true;
        }

        listObjectsReqManual = listObjectsReqManual.toBuilder()
            .continuationToken(listObjResponse.nextContinuationToken())
            .build();
    }
}

public static void anotherListExample(S3Client s3, String bucketName) {

    ListObjectsV2Request listReq = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

    // Process response pages
    listRes.stream()
        .flatMap(r -> r.contents().stream())
}
```

```
.forEach(content -> System.out.println(" Key: " + content.key() + " size = " + content.size()));

        // Helper method to work with paginated collection of items directly
        listRes.contents().stream()
            .forEach(content -> System.out.println(" Key: " + content.key() + " size = " + content.size()));

        for (S3Object content : listRes.contents()) {
            System.out.println(" Key: " + content.key() + " size = " + content.size());
        }
    }

    public static void deleteObjectFromBucket(S3Client s3, String bucketName,
String key) {

    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    s3.deleteObject(deleteObjectRequest);
    System.out.println(key +" was deleted");
}

public static String copyBucketObject (S3Client s3, String fromBucket, String
objectKey, String toBucket) {

    String encodedUrl = null;
    try {
        encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
StandardCharsets.UTF_8.toString());
    } catch (UnsupportedEncodingException e) {
        System.out.println("URL could not be encoded: " + e.getMessage());
    }
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .copySource(encodedUrl)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        System.out.println("The "+ objectKey +" was copied to "+toBucket);
        return copyRes.copyObjectResult().toString();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)

- [DeleteObjects](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import {  
    CreateBucketCommand,  
    PutObjectCommand,  
    CopyObjectCommand,  
    DeleteObjectCommand,  
    DeleteBucketCommand,  
    GetObjectCommand  
} from "@aws-sdk/client-s3";  
import { s3Client } from "../libs/s3Client.js"; // Helper function that creates an  
Amazon S3 service client module.  
  
if (process.argv.length < 5) {  
    console.log(  
        "Usage: node s3_basics.js <the bucket name> <the AWS Region to use> <object  
name> <object content>\n" +  
        "Example: node s3_basics_full.js test-bucket 'test.txt' 'Test Content'"  
    );  
}  
const bucket_name = process.argv[2];  
const object_key = process.argv[3];  
const object_content = process.argv[4];  
  
export const run = async (bucket_name, object_key, object_content) => {  
    try {  
        const create_bucket_params = {  
            Bucket: bucket_name  
        };  
        console.log("\nCreating the bucket, named " + bucket_name + "...\\n");  
        console.log("about to create");  
        const data = await s3Client.send(  
            new CreateBucketCommand(create_bucket_params)  
        );  
        console.log("Bucket created at ", data.Location);  
        try {  
            console.log(  
                "\nCreated and uploaded an object named " +  
                object_key +  
                " to first bucket " +  
                bucket_name +  
                " ...\\n"  
            );  
            // Set the parameters for the object to upload.  
            const object_upload_params = {  
                Bucket: bucket_name,  
                // Specify the name of the new object. For example, 'test.html'.  
                // To create a directory for the object, use '/'. For example, 'myApp/  
package.json'.  
                Key: object_key,  
                // Content of the new object.  
            };  
        } catch (err) {  
            console.error("Error creating or uploading object: " + err.message);  
        }  
    } catch (err) {  
        console.error("Error creating bucket: " + err.message);  
    }  
};
```

```
        Body: object_content,
    };
    // Create and upload the object to the first S3 bucket.
    await s3Client.send(new PutObjectCommand(object_upload_params));
    console.log(
        "Successfully uploaded object: " +
        object_upload_params.Bucket +
        "/" +
        object_upload_params.Key
    );
    try {
        const download_bucket_params = {
            Bucket: bucket_name,
            Key: object_key
        };
        console.log(
            "\nDownloading " +
            object_key +
            " from" +
            bucket_name +
            " ...\n"
        );
        // Create a helper function to convert a ReadableStream into a string.
        const streamToString = (stream) =>
            new Promise((resolve, reject) => {
                const chunks = [];
                stream.on("data", (chunk) => chunks.push(chunk));
                stream.on("error", reject);
                stream.on("end", () =>
                    resolve(Buffer.concat(chunks).toString("utf8")));
            });
        // Get the object from the Amazon S3 bucket. It is returned as a
        // ReadableStream.
        const data = await s3Client.send(new
        GetObjectCommand(download_bucket_params));
        // Convert the ReadableStream to a string.
        const bodyContents = await streamToString(data.Body);
        console.log(bodyContents);
        try {
            // Copy the object from the first bucket to the second bucket.
            const copy_object_params = {
                Bucket: bucket_name,
                CopySource: "/" + bucket_name + "/" + object_key,
                Key: "copy-destination/" + object_key,
            };
            console.log(
                "\nCopying " +
                object_key +
                " from" +
                bucket_name +
                " to " +
                bucket_name +
                "/" +
                copy_object_params.Key +
                " ...\n"
            );
            await s3Client.send(new CopyObjectCommand(copy_object_params));
            console.log("Success, object copied to folder.");
            try {
                console.log("\nDeleting " + object_key + " from" + bucket_name);
                const delete_object_from_bucket_params = {
                    Bucket: bucket_name,
                    Key: object_key,
                };
            }
        }
    }
}
```

```
        await s3Client.send(
            new DeleteObjectCommand(delete_object_from_bucket_params)
        );
        console.log("Success. Object deleted from bucket.");
        try {
            console.log(
                "\nDeleting " +
                object_key +
                " from " +
                bucket_name +
                "/copy-destination folder"
            );
            const delete_object_from_folder_params = {
                Bucket: bucket_name,
                Key: "copy-destination/" + object_key,
            };

            await s3Client.send(
                new DeleteObjectCommand(delete_object_from_folder_params)
            );
            console.log("Success. Object deleted from folder.");
            try {
                console.log(
                    "\nDeleting the bucket named " + bucket_name + "...\\n"
                );
                const delete_bucket_params = {Bucket: bucket_name};
                await s3Client.send(
                    new DeleteBucketCommand(delete_bucket_params)
                );
                console.log("Success. First bucket deleted.");
                return "Run successfully"; // For unit tests.
            } catch (err) {
                console.log("Error deleting object from folder.", err);
                process.exit(1);
            }
        } catch (err) {
            console.log("Error deleting bucket.", err);
            process.exit(1);
        }
    } catch (err) {
        console.log("Error deleting object from bucket.", err);
        process.exit(1);
    }
} catch (err) {
    console.log("Error copying object from to folder", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error downloading object", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error creating and upload object to bucket", err);
    process.exit(1);
}
console.log("works");
} catch (err) {
    console.log("Error creating bucket", err);
}
};

run(bucket_name, object_key, object_content);
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun main(args: Array<String>) {  
  
    val usage = """  
    Usage:  
        <bucketName> <key> <objectPath> <savePath> <toBucket>  
  
    Where:  
        bucketName - The Amazon S3 bucket to create.  
        key - The key to use.  
        objectPath - The path where the file is located (for example, C:/AWS/  
book2.pdf).  
        savePath - The path where the file is saved after it's downloaded (for  
example, C:/AWS/book2.pdf).  
        toBucket - An Amazon S3 bucket to where an object is copied to (for  
example, C:/AWS/book2.pdf).  
    """  
  
    if (args.size != 4) {  
        println(usage)  
        exitProcess(1)  
    }  
  
    val bucketName = args[0]  
    val key = args[1]  
    val objectPath = args[2]  
    val savePath = args[3]  
    val toBucket = args[4]  
  
    // Create an Amazon S3 bucket.  
    createBucket(bucketName)  
  
    // Update a local file to the Amazon S3 bucket.  
    putObject(bucketName, key, objectPath)  
  
    // Download the object to another local file.  
    getObject(bucketName, key, savePath)  
  
    // List all objects located in the Amazon S3 bucket.  
    listBucketObs(bucketName)
```

```
// Copy the object to another Amazon S3 bucket
copyBucket0b(bucketName, key, toBucket)

// Delete the object from the Amazon S3 bucket.
deleteBucket0bs(bucketName, key)

// Delete the Amazon S3 bucket.
deleteBucket(bucketName)
println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {

    val request = CreateBucketRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(bucketName: String, objectKey: String, objectPath: String) {

    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request = PutObjectRequest {
        bucket = bucketName
        key = objectKey
        metadata = metadataVal
        this.body = Paths.get(objectPath).asByteStream()
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObject(bucketName: String, keyName: String, path: String) {

    val request = GetObjectRequest {
        key = keyName
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeTo(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucket0bs(bucketName: String) {

    val request = ListObjectsRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
```

```

        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The owner is ${myObject.owner}")
        }
    }
}

suspend fun copyBucket0b(fromBucket: String, objectKey: String, toBucket: String) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        println("URL could not be encoded: " + e.message)
    }

    val request = CopyObjectRequest {
        copySource = encodedUrl
        bucket = toBucket
        key = objectKey
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucket0bs(bucketName: String, objectName: String) {
    val objectId = ObjectIdentifier {
        key = objectName
    }

    val del0b = Delete {
        objects = listOf(objectId)
    }

    val request = DeleteObjectsRequest {
        bucket = bucketName
        delete = del0b
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request = DeleteBucketRequest {
        bucket = bucketName
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}

```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.

- [CopyObject](#)
- [CreateBucket](#)

- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

PHP

## SDK for PHP

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;

echo("-----\n");
print("Welcome to the Amazon S3 getting started demo using PHP!\n");
echo("-----\n");

$region = 'us-west-2';
$version = 'latest';

$s3client = new S3Client([
    'region' => $region,
    'version' => $version
]);
/* Inline declaration example
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);
*/
$bucket_name = "doc-example-bucket-" . uniqid();

try {
    $s3client->createBucket([
        'Bucket' => $bucket_name,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
    echo "Created bucket named: $bucket_name \n";
} catch (Exception $exception) {
    echo "Failed to create bucket $bucket_name with error: " . $exception-
>getMessage();
    exit("Please fix error with bucket creation before continuing.");
}

$file_name = "local-file-" . uniqid();
try {
    $s3client->putObject([
        'Bucket' => $bucket_name,
        'Key' => $file_name,
        'SourceFile' => 'testfile.txt'
    ]);
    echo "Uploaded $file_name to $bucket_name.\n";
} catch (Exception $exception) {
    echo "Failed to upload $file_name with error: " . $exception->getMessage();
    exit("Please fix error with file upload before continuing.");
}

try {
```

```
$file = $s3client->getObject([
    'Bucket' => $bucket_name,
    'Key' => $file_name,
]);
$body = $file->get('Body');
$body->rewind();
echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
    echo "Failed to download $file_name from $bucket_name with error: " .
$exception->getMessage();
    exit("Please fix error with file downloading before continuing.");
}

try {
    $folder = "copied-folder";
    $s3client->copyObject([
        'Bucket' => $bucket_name,
        'CopySource' => "$bucket_name/$file_name",
        'Key' => "$folder/$file_name-copy",
    ]);
    echo "Copied $file_name to $folder/$file_name-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $file_name with error: " . $exception->getMessage();
    exit("Please fix error with object copying before continuing.");
}

try {
    $contents = $s3client->listObjects([
        'Bucket' => $bucket_name,
    ]);
    echo "The contents of your bucket are: \n";
    foreach ($contents['Contents'] as $content) {
        echo $content['Key'] . "\n";
    }
} catch (Exception $exception) {
    echo "Failed to list objects in $bucket_name with error: " . $exception-
>getMessage();
    exit("Please fix error with listing objects before continuing.");
}

try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [
            'Key' => $content['Key'],
        ];
    }
    $s3client->deleteObjects([
        'Bucket' => $bucket_name,
        'Key' => $file_name,
        'Delete' => [
            'Objects' => $objects,
        ],
    ]);
    $check = $s3client->listObjects([
        'Bucket' => $bucket_name,
    ]);
    if (count($check) <= 0) {
        throw new Exception("Bucket wasn't empty.");
    }
    echo "Deleted all objects and folders from $bucket_name.\n";
} catch (Exception $exception) {
    echo "Failed to delete $file_name from $bucket_name with error: " . $exception-
>getMessage();
    exit("Please fix error with object deletion before continuing.");
}
```

```
try {
    $s3client->deleteBucket([
        'Bucket' => $bucket_name,
    ]);
    echo "Deleted bucket $bucket_name.\n";
} catch (Exception $exception) {
    echo "Failed to delete $bucket_name with error: " . $exception->getMessage();
    exit("Please fix error with bucket deletion before continuing.");
}

echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- For API details, see the following topics in *AWS SDK for PHP API Reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import io
import os
import uuid

import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError


def do_scenario(s3_resource):
    print('*'*88)
    print("Welcome to the Amazon S3 getting started demo!")
    print('*'*88)

    bucket_name = f'doc-example-bucket-{uuid.uuid4()}' 
    bucket = s3_resource.Bucket(bucket_name)
    try:
        bucket.create(
            CreateBucketConfiguration={
                'LocationConstraint': s3_resource.meta.client.meta.region_name})
        print(f"Created demo bucket named {bucket.name}.")
    except ClientError as err:
        print(f" Tried and failed to create demo bucket {bucket_name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']['Message']}")
        print(f"\nCan't continue the demo without a bucket!")
    return
```

```
file_name = None
while file_name is None:
    file_name = input("\nEnter a file you want to upload to your bucket: ")
    if not os.path.exists(file_name):
        print(f"Couldn't find file {file_name}. Are you sure it exists?")
        file_name = None

obj = bucket.Object(os.path.basename(file_name))
try:
    obj.upload_file(file_name)
    print(f"Uploaded file {file_name} into bucket {bucket.name} with key {obj.key}.")
except S3UploadFailedError as err:
    print(f"Couldn't upload file {file_name} to {bucket.name}.")
    print(f"\t{err}")

answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
if answer.lower() == 'y':
    data = io.BytesIO()
    try:
        obj.download_fileobj(data)
        data.seek(0)
        print(f"Got your object. Here are the first 20 bytes:\n")
        print(f"\t{data.read(20)}")
    except ClientError as err:
        print(f"Couldn't download {obj.key}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']['Message']}")

    answer = input(
        f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? ")
    if answer.lower() == 'y':
        dest_obj = bucket.Object(f'demo-folder/{obj.key}')
        try:
            dest_obj.copy({'Bucket': bucket.name, 'Key': obj.key})
            print(f"Copied {obj.key} to {dest_obj.key}.")
        except ClientError as err:
            print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
            print(f"\t{err.response['Error']['Code']}:{err.response['Error']['Message']}")

print("\nYour bucket contains the following objects:")
try:
    for o in bucket.objects.all():
        print(f"\t{o.key}")
except ClientError as err:
    print(f"Couldn't list the objects in bucket {bucket.name}.")
    print(f"\t{err.response['Error']['Code']}:{err.response['Error']['Message']}")

answer = input(
    "\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
if answer.lower() == 'y':
    try:
        bucket.objects.delete()
        bucket.delete()
        print(f"Emptied and deleted bucket {bucket.name}.\n")
    except ClientError as err:
        print(f"Couldn't empty and delete bucket {bucket.name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']['Message']}")

print("Thanks for watching!")
print('-'*88)
```

```
if __name__ == '__main__':
    do_scenario(boto3.resource('s3'))
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "doc-example-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: "us-east-2" # Note: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
    rescue Aws::Errors::ServiceError => e
      puts("Tried and failed to create demo bucket.")
      puts("\t#{e.code}: #{e.message}")
      puts("\nCan't continue the demo without a bucket!")
      raise
    else
      bucket
    end

    # Requests a file name from the user.
    #
    # @return The name of the file.
    def create_file
      File.open("demo.txt", w) { |f| f.write("This is a demo file.") }
    end
  end
end
```

```
end

# Uploads a file to an Amazon S3 bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket object representing the upload
# destination
# @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded file.
def upload_file(bucket)
  File.open("demo.txt", "w+") { |f| f.write("This is a demo file.") }
  s3_object = bucket.object(File.basename("demo.txt"))
  s3_object.upload_file("demo.txt")
  puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't upload file demo.txt to #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  s3_object
end

# Downloads an Amazon S3 object to a file.
#
# @param s3_object [Aws::S3::Object] The object to download.
def download_file(s3_object)
  puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
  answer = gets.chomp.downcase
  if answer == "y"
    puts("Enter a name for the downloaded file: ")
    file_name = gets.chomp
    s3_object.download_file(file_name)
    puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't download #{s3_object.key}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Copies an Amazon S3 object to a subfolder within the same bucket.
#
# @param source_object [Aws::S3::Object] The source object to copy.
# @return [Aws::S3::Object, nil] The destination object.
def copy_object(source_object)
  dest_object = nil
  puts("\nDo you want to copy #{source_object.key} to a subfolder in your bucket
(y/n)? ")
  answer = gets.chomp.downcase
  if answer == "y"
    dest_object = source_object.bucket.object("demo-folder/#{source_object.key}")
    dest_object.copy_from(source_object)
    puts("Copied #{source_object.key} to #{dest_object.key}.")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't copy #{source_object.key}.")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  dest_object
end

# Lists the objects in an Amazon S3 bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to query.
def list_objects(bucket)
  puts("\nYour bucket contains the following objects:")
```

```
bucket.objects.each do |obj|
    puts("\t#{obj.key}")
end
rescue Aws::Errors::ServiceError => e
    puts("Couldn't list the objects in bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
end

# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")
    answer = gets.chomp.downcase
    if answer == "y"
        bucket.objects.batch_delete!
        bucket.delete
        puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
end
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
    puts("-" * 88)
    puts("Welcome to the Amazon S3 getting started demo!")
    puts("-" * 88)

    bucket = scenario.create_bucket
    s3_object = scenario.upload_file(bucket)
    scenario.download_file(s3_object)
    scenario.copy_object(s3_object)
    scenario.list_objects(bucket)
    scenario.delete_bucket(bucket)

    puts("Thanks for watching!")
    puts("-" * 88)
rescue Aws::Errors::ServiceError
    puts("Something went wrong with the demo!")
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME ==
--FILE--
```

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

Rust

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the binary crate which runs the scenario.

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::{Client, Error, Region};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), Error> {
    let (region, client, bucket_name, file_name, key, target_key) =
        initialize_variables().await;

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name, key, target_key).await
    {
        println!("{}: {}", e);
    }

    Ok(())
}

async fn initialize_variables() -> (Region, Client, String, String, String, String)
{
    let region_provider = RegionProviderChain::first_try(Region::new("us-west-2"));
    let region = region_provider.region().await.unwrap();

    let shared_config =
        aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);

    let bucket_name = format!("doc-example-bucket-{}", Uuid::new_v4());

    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
    let target_key = "target_key".to_string();

    (region, client, bucket_name, file_name, key, target_key)
}

async fn run_s3_operations(
    region: Region,
    client: Client,
    bucket_name: String,
    file_name: String,
    key: String,
    target_key: String,
) -> Result<(), Error> {
    s3_service::create_bucket(&client, &bucket_name, region.as_ref()).await?;
    s3_service::upload_object(&client, &bucket_name, &file_name, &key).await?;
    let _object = s3_service::download_object(&client, &bucket_name, &key).await;
    s3_service::copy_object(&client, &bucket_name, &key, &target_key).await?;
}
```

```
s3_service::list_objects(&client, &bucket_name).await?;
s3_service::delete_objects(&client, &bucket_name).await?;
s3_service::delete_bucket(&client, &bucket_name).await?;

Ok(())
}
```

A library crate with common actions called by the binary.

```
use aws_sdk_s3::model::{
    BucketLocationConstraint, CreateBucketConfiguration, Delete, ObjectIdentifier,
};
use aws_sdk_s3::output::{GetObjectOutput, ListObjectsV2Output};
use aws_sdk_s3::types::ByteStream;
use aws_sdk_s3::{Client, Error};
use std::path::Path;
use std::str;

pub async fn delete_bucket(client: &Client, bucket_name: &str) -> Result<(), Error> {
    client.delete_bucket().bucket(bucket_name).send().await?;
    println!("Bucket deleted");
    Ok(())
}

pub async fn delete_objects(client: &Client, bucket_name: &str) -> Result<(), Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    let mut delete_objects: Vec<ObjectIdentifier> = vec![];
    for obj in objects.contents().unwrap_or_default() {
        let obj_id = ObjectIdentifier::builder()
            .set_key(Some(obj.key().unwrap().to_string()))
            .build();
        delete_objects.push(obj_id);
    }
    client
        .delete_objects()
        .bucket(bucket_name)
        .delete(Delete::builder().set_objects(Some(delete_objects)).build())
        .send()
        .await?;

    let objects: ListObjectsV2Output =
        client.list_objects_v2().bucket(bucket_name).send().await?;
    match objects.key_count {
        0 => Ok(()),
        _ => Err(Error::Unhandled(Box::from(
            "There were still objects left in the bucket.",
        ))),
    }
}

pub async fn list_objects(client: &Client, bucket_name: &str) -> Result<(), Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;
    println!("Objects in bucket:");
    for obj in objects.contents().unwrap_or_default() {
        println!("{}:{}", obj.key().unwrap());
    }
}

Ok()
```

```
}

pub async fn copy_object(
    client: &Client,
    bucket_name: &str,
    object_key: &str,
    target_key: &str,
) -> Result<(), Error> {
    let mut source_bucket_and_object: String = "".to_owned();
    source_bucket_and_object.push_str(bucket_name);
    source_bucket_and_object.push('/');
    source_bucket_and_object.push_str(object_key);

    client
        .copy_object()
        .copy_source(source_bucket_and_object)
        .bucket(bucket_name)
        .key(target_key)
        .send()
        .await?;

    Ok(())
}

pub async fn download_object(client: &Client, bucket_name: &str, key: &str) ->
GetObjectOutput {
    let resp = client
        .get_object()
        .bucket(bucket_name)
        .key(key)
        .send()
        .await;
    resp.unwrap()
}

pub async fn upload_object(
    client: &Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<(), Error> {
    let body = ByteStream::from_path(Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await?;

    println!("Uploaded file: {}", file_name);
    Ok(())
}

pub async fn create_bucket(client: &Client, bucket_name: &str, region: &str) ->
Result<(), Error> {
    let constraint = BucketLocationConstraint::from(region);
    let cfg = CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await?;
```

```
    println!("Creating bucket named: {bucket_name}");
    Ok(())
}
```

- For API details, see the following topics in *AWS SDK for Rust API reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

" Create S3 Bucket "
TRY.
    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
    ).
    MESSAGE 'S3 bucket created' TYPE 'I'.
    CATCH /aws1/cx_s3_bucketalrdyexists.
        MESSAGE 'Bucket name already exists' TYPE 'E'.
        CATCH /aws1/cx_s3_bktalrdyownedbyyou.
            MESSAGE 'Bucket already exist and is owned by you' TYPE 'E'.
ENDTRY.

"Upload an object to a S3 bucket"
TRY.
    "Get contents of file from application server"
    DATA lv_file_content TYPE xstring.
    OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
    READ DATASET iv_key INTO lv_file_content.
    CLOSE DATASET iv_key.

    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_file_content
    ).
    MESSAGE 'Object uploaded to S3 bucket' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist' TYPE 'E'.
ENDTRY.
```

```
" Get an object from a bucket "
TRY.
    DATA(lo_result) = lo_s3->getobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
    ).
    DATA(lv_object_data) = lo_result->get_body( ).
    MESSAGE 'Object retrieved from S3 bucket' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist' TYPE 'E'.
    CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist' TYPE 'E'.
ENDTRY.

" Copy an object to a subfolder in a bucket "
TRY.
    lo_s3->copyobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }|
        iv_copysource = |{ iv_bucket_name }/{ iv_key }|
    ).
    MESSAGE 'Object copied to a subfolder' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist' TYPE 'E'.
    CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist' TYPE 'E'.
ENDTRY.

" List objects in the bucket "
TRY.
    DATA(lo_list) = lo_s3->listobjects(
        iv_bucket = iv_bucket_name
    ).
    MESSAGE 'Retrieved list of object(s) in S3 bucket' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist' TYPE 'E'.
ENDTRY.
DATA text TYPE string VALUE 'Object List - '.
DATA lv_object_key TYPE /aws1/s3_objectkey.
LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
    lv_object_key = lo_object->get_key( ).
    CONCATENATE lv_object_key ', ' INTO text.
ENDLOOP.
MESSAGE text TYPE 'I'.

" Delete the objects in a bucket "
TRY.
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
    ).
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }|
    ).
    MESSAGE 'Object(s) deleted from S3 bucket' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist' TYPE 'E'.
ENDTRY.

" Delete the bucket "
TRY.
    lo_s3->deletebucket(
        iv_bucket = iv_bucket_name
    ).
```

```
MESSAGE 'Deleted S3 bucket' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
MESSAGE 'Bucket does not exist' TYPE 'E'.
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

## Swift

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

A Swift class that handles calls to the SDK for Swift.

```
import Foundation
import AWSS3
import ClientRuntime
import AWSClientRuntime

/// A class containing all the code that interacts with the AWS SDK for Swift.
public class ServiceHandler {
    let client: S3Client

    /// Initialize and return a new ``ServiceHandler`` object, which is used to
    /// drive the AWS calls
    /// used for the example.
    ///
    /// - Returns: A new ``ServiceHandler`` object, ready to be called to
    /// execute AWS operations.
    public init() async {
        do {
            client = try await S3Client()
        } catch {
            print("ERROR: ", dump(error, name: "Initializing s3 client"))
            exit(1)
        }
    }

    /// Create a new user given the specified name.
    ///
    /// - Parameters:
    ///   - name: Name of the bucket to create.
    ///   - Throws an exception if an error occurs.
    public func createBucket(name: String) async throws {
        let config = S3ClientTypes.CreateBucketConfiguration(
            locationConstraint: .usEast2
        )
    }
}
```

```
let input = CreateBucketInput(  
    bucket: name,  
    createBucketConfiguration: config  
)  
    - = try await client.createBucket(input: input)  
}  
  
/// Delete a bucket.  
/// - Parameter name: Name of the bucket to delete.  
public func deleteBucket(name: String) async throws {  
    let input = DeleteBucketInput(  
        bucket: name  
)  
    - = try await client.deleteBucket(input: input)  
}  
  
/// Upload a file from local storage to the bucket.  
/// - Parameters:  
///   - bucket: Name of the bucket to upload the file to.  
///   - key: Name of the file to create.  
///   - file: Path name of the file to upload.  
public func uploadFile(bucket: String, key: String, file: String) async throws {  
    let urlString = URL(fileURLWithPath: file)  
    let fileData = try Data(contentsOf: urlString)  
    let dataStream = ByteStream.from(data: fileData)  
  
    let input = PutObjectInput(  
        body: dataStream,  
        bucket: bucket,  
        key: key  
)  
    - = try await client.putObject(input: input)  
}  
  
/// Create a file in the specified bucket with the given name. The new  
/// file's contents are uploaded from a `Data` object.  
///  
/// - Parameters:  
///   - bucket: Name of the bucket to create a file in.  
///   - key: Name of the file to create.  
///   - data: A `Data` object to write into the new file.  
public func createFile(bucket: String, key: String, withData data: Data) async throws {  
    let dataStream = ByteStream.from(data: data)  
  
    let input = PutObjectInput(  
        body: dataStream,  
        bucket: bucket,  
        key: key  
)  
    - = try await client.putObject(input: input)  
}  
  
/// Download the named file to the given directory on the local device.  
///  
/// - Parameters:  
///   - bucket: Name of the bucket that contains the file to be copied.  
///   - key: The name of the file to copy from the bucket.  
///   - to: The path of the directory on the local device where you want to  
///     download the file.  
public func downloadFile(bucket: String, key: String, to: String) async throws {  
    let urlString = URL(fileURLWithPath: to).appendingPathComponent(key)  
  
    let input = GetObjectInput(  
        bucket: bucket,  
        key: key  
)  
    - = try await client.getObject(input: input)  
}
```

```
        bucket: bucket,
        key: key
    )
    let output = try await client.getObject(input: input)

    // Get the data stream object. Return immediately if there isn't one.
    guard let body = output.body else {
        return
    }
    let data = body.toBytes().toData()
    try data.write(to: fileUrl)
}

/// Read the specified file from the given S3 bucket into a Swift
/// `Data` object.
///
/// - Parameters:
///   - bucket: Name of the bucket containing the file to read.
///   - key: Name of the file within the bucket to read.
///
/// - Returns: A `Data` object containing the complete file data.
public func readFile(bucket: String, key: String) async throws -> Data {
    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    let output = try await client.getObject(input: input)

    // Get the stream and return its contents in a `Data` object. If
    // there is no stream, return an empty `Data` object instead.
    guard let body = output.body else {
        return "".data(using: .utf8)!
    }
    let data = body.toBytes().toData()
    return data
}

/// Copy a file from one bucket to another.
///
/// - Parameters:
///   - sourceBucket: Name of the bucket containing the source file.
///   - name: Name of the source file.
///   - destBucket: Name of the bucket to copy the file into.
    public func copyFile(from sourceBucket: String, name: String, to destBucket:
String) async throws {
        let srcUrl = ("\"sourceBucket)/
\"name\"").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        _ = try await client.copyObject(input: input)
    }

    /// Deletes the specified file from Amazon S3.
    ///
    /// - Parameters:
    ///   - bucket: Name of the bucket containing the file to delete.
    ///   - key: Name of the file to delete.
    ///
    public func deleteFile(bucket: String, key: String) async throws {
        let input = DeleteObjectInput(
            bucket: bucket,
            key: key
    }
```

```
)  
  
    do {  
        _ = try await client.deleteObject(input: input)  
    } catch {  
        throw error  
    }  
}  
  
/// Returns an array of strings, each naming one file in the  
/// specified bucket.  
///  
/// - Parameter bucket: Name of the bucket to get a file listing for.  
/// - Returns: An array of `String` objects, each giving the name of  
///             one file contained in the bucket.  
public func listBucketFiles(bucket: String) async throws -> [String] {  
    let input = ListObjectsV2Input(  
        bucket: bucket  
    )  
    let output = try await client.listObjectsV2(input: input)  
    var names: [String] = []  
  
    guard let objList = output.contents else {  
        return []  
    }  
  
    for obj in objList {  
        if let objName = obj.key {  
            names.append(objName)  
        }  
    }  
  
    return names  
}
```

A Swift command-line program to manage the SDK calls.

```
import Foundation  
import ServiceHandler  
import ArgumentParser  
  
/// The command-line arguments and options available for this  
/// example command.  
struct ExampleCommand: ParsableCommand {  
    @Argument(help: "Name of the S3 bucket to create")  
    var bucketName: String  
  
    @Argument(help: "Pathname of the file to upload to the S3 bucket")  
    var uploadSource: String  
  
    @Argument(help: "The name (key) to give the file in the S3 bucket")  
    var objName: String  
  
    @Argument(help: "S3 bucket to copy the object to")  
    var destBucket: String  
  
    @Argument(help: "Directory where you want to download the file from the S3  
    bucket")  
    var downloadDir: String  
  
    static var configuration = CommandConfiguration(  
        commandName: "s3-basics",  
        abstract: "Demonstrates a series of basic AWS S3 functions.",  
    )
```

```
discussion: """
Performs the following Amazon S3 commands:

* `CreateBucket`
* `PutObject`
* `GetObject`
* `CopyObject`
* `ListObjects`
* `DeleteObjects`
* `DeleteBucket`
"""

)

/// Called by ``main()`` to do the actual running of the AWS
/// example.
func runAsync() async throws {
    let serviceHandler = await ServiceHandler()

    // 1. Create the bucket.
    print("Creating the bucket \(bucketName)...")
    try await serviceHandler.createBucket(name: bucketName)

    // 2. Upload a file to the bucket.
    print("Uploading the file \(uploadSource)...")
    try await serviceHandler.uploadFile(bucket: bucketName, key: objName, file: uploadSource)

    // 3. Download the file.
    print("Downloading the file \(objName) to \(downloadDir)...")
    try await serviceHandler.downloadFile(bucket: bucketName, key: objName, to: downloadDir)

    // 4. Copy the file to another bucket.
    print("Copying the file to the bucket \(destBucket)...")
    try await serviceHandler.copyFile(from: bucketName, name: objName, to: destBucket)

    // 5. List the contents of the bucket.

    print("Getting a list of the files in the bucket \(bucketName)")
    let fileList = try await serviceHandler.listBucketFiles(bucket: bucketName)
    let numFiles = fileList.count
    if numFiles != 0 {
        print("\(numFiles) file\(((numFiles > 1) ? "s" : "") in bucket
\(bucketName):")
        for name in fileList {
            print("  \(name)")
        }
    } else {
        print("No files found in bucket \(bucketName)")
    }

    // 6. Delete the objects from the bucket.

    print("Deleting the file \(objName) from the bucket \(bucketName)...")
    try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
    print("Deleting the file \(objName) from the bucket \(destBucket)...")
    try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

    // 7. Delete the bucket.
    print("Deleting the bucket \(bucketName)...")
    try await serviceHandler.deleteBucket(name: bucketName)

    print("Done.")
}

}
```

```
//  
// Main program entry point.  
//  
@main  
struct Main {  
    static func main() async {  
        let args = Array(CommandLine.arguments.dropFirst())  
  
        do {  
            let command = try ExampleCommand.parse(args)  
            try await command.runAsync()  
        } catch {  
            ExampleCommand.exit(withError: error)  
        }  
    }  
}
```

- For API details, see the following topics in *AWS SDK for Swift API reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

## Manage versioned Amazon S3 objects in batches with a Lambda function using an AWS SDK

The following code example shows how to manage versioned S3 objects in batches with a Lambda function.

Python

### SDK for Python (Boto3)

Shows how to manipulate Amazon Simple Storage Service (Amazon S3) versioned objects in batches by creating jobs that call AWS Lambda functions to perform processing. This example creates a version-enabled bucket, uploads the stanzas from the poem *You Are Old, Father William* by Lewis Carroll, and uses Amazon S3 batch jobs to twist the poem in various ways.

#### Learn how to:

- Create Lambda functions that operate on versioned objects.
- Create a manifest of objects to update.
- Create batch jobs that invoke Lambda functions to update objects.
- Delete Lambda functions.
- Empty and delete a versioned bucket.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon S3

## Upload or download large files to and from Amazon S3 using an AWS SDK

The following code examples show how to upload or download large files to and from Amazon S3.

For more information, see [Uploading an object using multipart upload](#).

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Call functions that transfer files to and from an S3 bucket using the Amazon S3 TransferUtility.

```
global using System.Text;
global using Amazon;
global using Amazon.S3;
global using Amazon.S3.Model;
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;

// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
    $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\\
    \TransferFolder";

DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");
var success = await TransferMethods.UploadSingleFileAsync(transferUtil, bucketName,
    fileToUpload, localPath);
```

```
if (success)
{
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
{bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil, bucketName,
keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
{bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDelete"];
Console.WriteLine($"Downloading {keyName} from {bucketName}.");

success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
keyName, localPath);
if (success)
{
    Console.WriteLine($"Successfully downloaded the file, {keyName} from
{bucketName}.");
}

PressEnter();

// Download the contents of a directory from an S3 bucket.
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
var downloadPath = $"{localPath}\\\{s3Path}";

Console.WriteLine($"Downloading the contents of {bucketName}\\\{s3Path}");
Console.WriteLine($"{bucketName}\\\{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();

success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil, bucketName,
s3Path, downloadPath);
if (success)
{
```

```
        Console.WriteLine($"Downloaded the files in {bucketName} to {downloadPath}.");
        Console.WriteLine($"{downloadPath} now contains the following files:");
        DisplayLocalFiles(downloadPath);
    }

    Console.WriteLine("\nThe TransferUtility Basics application has completed.");
    PressEnter();

    // Displays the title for a section of the scenario.
    static void DisplayTitle(string titleText)
    {
        var sepBar = new string('-', Console.WindowWidth);

        Console.WriteLine(sepBar);
        Console.WriteLine(CenterText(titleText));
        Console.WriteLine(sepBar);
    }

    // Displays a description of the actions to be performed by the scenario.
    static void DisplayInstructions()
    {
        var sepBar = new string('-', Console.WindowWidth);

        DisplayTitle("Amazon S3 Transfer Utility Basics");
        Console.WriteLine("This program shows how to use the Amazon S3 Transfer Utility.");
        Console.WriteLine("It performs the following actions:");
        Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
        Console.WriteLine("\t2. Upload an entire directory from the local computer to an\n\t\tS3 bucket.");
        Console.WriteLine("\t3. Download a single object from an S3 bucket.");
        Console.WriteLine("\t4. Download the objects in an S3 bucket to a local directory.");
        Console.WriteLine($"{sepBar}");
    }

    // Pauses the scenario.
    static void PressEnter()
    {
        Console.WriteLine("Press <Enter> to continue.");
        _ = Console.ReadLine();
        Console.WriteLine("\n");
    }

    // Returns the string textToCenter, padded on the left with spaces
    // that center the text on the console display.
    static string CenterText(string textToCenter)
    {
        var centeredText = new StringBuilder();
        var screenWidth = Console.WindowWidth;
        centeredText.Append(new string(' ', (int)(screenWidth - textToCenter.Length) / 2));
        centeredText.Append(textToCenter);
        return centeredText.ToString();
    }

    // Displays a list of file names included in the specified path.
    static void DisplayLocalFiles(string localPath)
    {
        var fileList = Directory.GetFiles(localPath);
        if (fileList.Length > 0)
        {
            foreach (var fileName in fileList)
            {
                Console.WriteLine(fileName);
            }
        }
    }
}
```

```
        }

    // Displays a list of the files in the specified S3 bucket and prefix.
    static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
    s3Path)
    {
        ListObjectsV2Request request = new()
        {
            BucketName = bucketName,
            Prefix = s3Path,
            MaxKeys = 5,
        };

        var response = new ListObjectsV2Response();

        do
        {
            response = await client.ListObjectsV2Async(request);

            response.S3Objects
                .ForEach(obj => Console.WriteLine($"'{obj.Key}'"));

            // If the response is truncated, set the request ContinuationToken
            // from the NextContinuationToken property of the response.
            request.ContinuationToken = response.NextContinuationToken;
        } while (response.IsTruncated);
    }
}
```

Upload a single file.

```
/// <summary>
/// Uploads a single file from the local computer to an S3 bucket.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
/// <param name="bucketName">The name of the S3 bucket where the file
/// will be stored.</param>
/// <param name="fileName">The name of the file to upload.</param>
/// <param name="localPath">The local path where the file is stored.</
param>
/// <returns>A boolean value indicating the success of the action.</
returns>
public static async Task<bool> UploadSingleFileAsync(
    TransferUtility transferUtil,
    string bucketName,
    string fileName,
    string localPath)
{
    if (File.Exists($"{localPath}\\{fileName}"))
    {
        try
        {
            await transferUtil.UploadAsync(new TransferUtilityUploadRequest
            {
                BucketName = bucketName,
                Key = fileName,
                FilePath = $"{localPath}\\{fileName}",
            });

            return true;
        }
    }
}
```

```
        catch (AmazonS3Exception s3Ex)
        {
            Console.WriteLine($"Could not upload {fileName} from
{localPath} because:");
            Console.WriteLine(s3Ex.Message);
            return false;
        }
    }
    else
    {
        Console.WriteLine($"{fileName} does not exist in {localPath}");
        return false;
    }
}
```

Upload an entire local directory.

```
///<summary>
/// Uploads all the files in a local directory to a directory in an S3
/// bucket.
///</summary>
///<param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
///<param name="bucketName">The name of the S3 bucket where the files
/// will be stored.</param>
///<param name="keyPrefix">The key prefix is the S3 directory where
/// the files will be stored.</param>
///<param name="localPath">The local directory that contains the files
/// to be uploaded.</param>
///<returns>A Boolean value representing the success of the action.</returns>
public static async Task<bool> UploadFullDirectoryAsync(
    TransferUtility transferUtil,
    string bucketName,
    string keyPrefix,
    string localPath)
{
    if (Directory.Exists(localPath))
    {
        try
        {
            await transferUtil.UploadDirectoryAsync(new
TransferUtilityUploadDirectoryRequest
            {
                BucketName = bucketName,
                KeyPrefix = keyPrefix,
                Directory = localPath,
            });

            return true;
        }
        catch (AmazonS3Exception s3Ex)
        {
            Console.WriteLine($"Can't upload the contents of {localPath}
because:");
            Console.WriteLine(s3Ex?.Message);
            return false;
        }
    }
    else
    {
        Console.WriteLine($"The directory {localPath} does not exist.");
        return false;
    }
}
```

```
    }  
}
```

Download a single file.

```
/// <summary>  
/// Download a single file from an S3 bucket to the local computer.  
/// </summary>  
/// <param name="transferUtil">The transfer initialized TransferUtility  
/// object.</param>  
/// <param name="bucketName">The name of the S3 bucket containing the  
/// file to download.</param>  
/// <param name="keyName">The name of the file to download.</param>  
/// <param name="localPath">The path on the local computer where the  
/// downloaded file will be saved.</param>  
/// <returns>A Boolean value indicating the results of the action.</returns>  
public static async Task<bool> DownloadSingleFileAsync(  
    TransferUtility transferUtil,  
    string bucketName,  
    string keyName,  
    string localPath)  
{  
    await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest  
    {  
        BucketName = bucketName,  
        Key = keyName,  
        FilePath = $"{localPath}\\{keyName}",  
    });  
  
    return (File.Exists($"{localPath}\\{keyName}"));  
}
```

Download contents of an S3 bucket.

```
/// <summary>  
/// Downloads the contents of a directory in an S3 bucket to a  
/// directory on the local computer.  
/// </summary>  
/// <param name="transferUtil">The transfer initialized TransferUtility  
/// object.</param>  
/// <param name="bucketName">The bucket containing the files to download.</param>  
/// <param name="s3Path">The S3 directory where the files are located.</param>  
/// <param name="localPath">The local path to which the files will be  
/// saved.</param>  
/// <returns>A Boolean value representing the success of the action.</returns>  
public static async Task<bool> DownloadS3DirectoryAsync(  
    TransferUtility transferUtil,  
    string bucketName,  
    string s3Path,  
    string localPath)  
{  
    int fileCount = 0;
```

```
// If the directory doesn't exist, it will be created.  
if (Directory.Exists(s3Path))  
{  
    var files = Directory.GetFiles(localPath);  
    fileCount = files.Length;  
}  
  
await transferUtil.DownloadDirectoryAsync(new  
TransferUtilityDownloadDirectoryRequest  
{  
    BucketName = bucketName,  
    LocalDirectory = localPath,  
    S3Directory = s3Path,  
});  
  
if (Directory.Exists(localPath))  
{  
    var files = Directory.GetFiles(localPath);  
    if (files.Length > fileCount)  
    {  
        return true;  
    }  
  
    // No change in the number of files. Assume  
    // the download failed.  
    return false;  
}  
  
// The local directory doesn't exist. No files  
// were downloaded.  
return false;  
}
```

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that transfer files using several of the available transfer manager settings. Use a callback class to write callback progress during file transfer.

```
import sys  
import threading  
  
import boto3  
from boto3.s3.transfer import TransferConfig  
  
MB = 1024 * 1024  
s3 = boto3.resource('s3')  
  
class TransferCallback:  
    """  
    Handle callbacks from the transfer manager.  
    """  
  
    The transfer manager periodically calls the __call__ method throughout  
    the upload and download process so that it can take action, such as
```

```
displaying progress to the user and collecting data about the transfer.  
"""  
  
def __init__(self, target_size):  
    self._target_size = target_size  
    self._total_transferred = 0  
    self._lock = threading.Lock()  
    self.thread_info = {}  
  
def __call__(self, bytes_transferred):  
    """  
        The callback method that is called by the transfer manager.  
  
        Display progress during file transfer and collect per-thread transfer  
        data. This method can be called by multiple threads, so shared instance  
        data is protected by a thread lock.  
    """  
    thread = threading.current_thread()  
    with self._lock:  
        self._total_transferred += bytes_transferred  
        if thread.ident not in self.thread_info.keys():  
            self.thread_info[thread.ident] = bytes_transferred  
        else:  
            self.thread_info[thread.ident] += bytes_transferred  
  
    target = self._target_size * MB  
    sys.stdout.write(  
        f"\r{self._total_transferred} of {target} transferred "  
        f"({(self._total_transferred / target) * 100:.2f}%).")  
    sys.stdout.flush()  
  
  
def upload_with_default_configuration(local_file_path, bucket_name,  
                                      object_key, file_size_mb):  
    """  
        Upload a file from a local folder to an Amazon S3 bucket, using the default  
        configuration.  
    """  
    transfer_callback = TransferCallback(file_size_mb)  
    s3.Bucket(bucket_name).upload_file(  
        local_file_path,  
        object_key,  
        Callback=transfer_callback)  
    return transfer_callback.thread_info  
  
  
def upload_with_chunksize_and_meta(local_file_path, bucket_name, object_key,  
                                    file_size_mb, metadata=None):  
    """  
        Upload a file from a local folder to an Amazon S3 bucket, setting a  
        multipart chunk size and adding metadata to the Amazon S3 object.  
  
        The multipart chunk size controls the size of the chunks of data that are  
        sent in the request. A smaller chunk size typically results in the transfer  
        manager using more threads for the upload.  
  
        The metadata is a set of key-value pairs that are stored with the object  
        in Amazon S3.  
    """  
    transfer_callback = TransferCallback(file_size_mb)  
  
    config = TransferConfig(multipart_chunksize=1 * MB)  
    extra_args = {'Metadata': metadata} if metadata else None  
    s3.Bucket(bucket_name).upload_file(  
        local_file_path,  
        object_key,
```

```
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback)
    return transfer_callback.thread_info

def upload_with_high_threshold(local_file_path, bucket_name, object_key,
                               file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        Callback=transfer_callback)
    return transfer_callback.thread_info

def upload_with_sse(local_file_path, bucket_name, object_key,
                    file_size_mb, sse_key=None):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {
            'SSECustomerAlgorithm': 'AES256',
            'SSECustomerKey': sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        ExtraArgs=extra_args,
        Callback=transfer_callback)
    return transfer_callback.thread_info

def download_with_default_configuration(bucket_name, object_key,
                                         download_file_path, file_size_mb):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path,
        Callback=transfer_callback)
    return transfer_callback.thread_info

def download_with_single_thread(bucket_name, object_key,
                                download_file_path, file_size_mb):
```

```
"""
Download a file from an Amazon S3 bucket to a local folder, using a
single thread.
"""
transfer_callback = TransferCallback(file_size_mb)
config = TransferConfig(use_threads=False)
s3.Bucket(bucket_name).Object(object_key).download_file(
    download_file_path,
    Config=config,
    Callback=transfer_callback)
return transfer_callback.thread_info

def download_with_high_threshold(bucket_name, object_key,
                                  download_file_path, file_size_mb):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path,
        Config=config,
        Callback=transfer_callback)
    return transfer_callback.thread_info

def download_with_sse(bucket_name, object_key, download_file_path,
                      file_size_mb, sse_key):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {
            'SSECustomerAlgorithm': 'AES256',
            'SSECustomerKey': sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path,
        ExtraArgs=extra_args,
        Callback=transfer_callback)
    return transfer_callback.thread_info
```

Demonstrate the transfer manager functions and report results.

```
import hashlib
import os
import platform
import shutil
import time
```

```
import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = ('multipart_threshold', 'multipart_chunksize', 'max_concurrency',
                 'use_threads')
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ('max_io_queue', 'io_chunksize', 'num_download_attempts')

class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource('s3')
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store "
                "demonstration files? ")
            if not os.path.isdir(self.demo_folder):
                print(f"{self.demo_folder} isn't a folder!")
                self.demo_folder = None

        while not self.demo_bucket:
            self.demo_bucket = input(
                "Which Amazon S3 bucket do you want to use to store "
                "demonstration files? ")
        try:
            self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
        except ParamValidationError as err:
            print(err)
            self.demo_bucket = None
        except ClientError as err:
            print(err)
            print(
                f"Either {self.demo_bucket} doesn't exist or you don't "
                f"have access to it.")
            self.demo_bucket = None

    def demo(self, question, upload_func, download_func,
            upload_args=None, download_args=None):
        """Run a demonstration.
```

Ask the user if they want to run this specific demonstration.  
If they say yes, create a file on the local path, upload it  
using the specified upload function, then download it using the  
specified download function.

```
"""
if download_args is None:
    download_args = {}
if upload_args is None:
    upload_args = {}
question = question.format(self.file_size_mb)
answer = input(f"{question} (y/n)")
if answer.lower() == 'y':
    local_file_path, object_key, download_file_path = \
        self._create_demo_file()

    file_transfer.TransferConfig = \
        self._config_wrapper(TransferConfig, CONFIG_ATTRS)
    self._report_transfer_params('Uploading', local_file_path,
                                 object_key, **upload_args)
    start_time = time.perf_counter()
    thread_info = upload_func(local_file_path, self.demo_bucket,
                             object_key, self.file_size_mb,
                             **upload_args)
    end_time = time.perf_counter()
    self._report_transfer_result(thread_info, end_time - start_time)

    file_transfer.TransferConfig = \
        self._config_wrapper(TransferConfig,
                           CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS)
    self._report_transfer_params('Downloading', object_key,
                                 download_file_path, **download_args)
    start_time = time.perf_counter()
    thread_info = download_func(self.demo_bucket, object_key,
                               download_file_path, self.file_size_mb,
                               **download_args)
    end_time = time.perf_counter()
    self._report_transfer_result(thread_info, end_time - start_time)

def last_name_set(self):
    """Get the name set used for the last demo."""
    return self._chore_list[-1]

def cleanup(self):
    """
    Remove files from the demo folder, and uploaded objects from the
    Amazon S3 bucket.
    """
    print('-' * self._terminal_width)
    for local_file_path, s3_object_key, downloaded_file_path \
            in self._chore_list:
        print(f"Removing {local_file_path}")
        try:
            os.remove(local_file_path)
        except FileNotFoundError as err:
            print(err)

        print(f"Removing {downloaded_file_path}")
        try:
            os.remove(downloaded_file_path)
        except FileNotFoundError as err:
            print(err)

    if self.demo_bucket:
        print(f"Removing {self.demo_bucket}:{s3_object_key}")
        try:
```

```
        self._s3.Bucket(self.demo_bucket).Object(
            s3_object_key).delete()
    except ClientError as err:
        print(err)

    def _setup_platform_specific(self):
        """Set up platform-specific command used to create a large file."""
        if platform.system() == "Windows":
            self._create_file_cmd = "fsutil file createnew {} {}"
            self._size_multiplier = MB
        elif platform.system() == "Linux" or platform.system() == "Darwin":
            self._create_file_cmd = f"dd if=/dev/urandom of={} " \
                f"bs={MB} count={{}"
            self._size_multiplier = 1
        else:
            raise EnvironmentError(
                f"Demo of platform {platform.system()} isn't supported.")

    def _create_demo_file(self):
        """
        Create a file in the demo folder specified by the user. Store the local
        path, object name, and download path for later cleanup.

        Only the local file is created by this method. The Amazon S3 object and
        download file are created later during the demonstration.

        Returns:
        A tuple that contains the local file path, object name, and download
        file path.
        """
        file_name_template = "TestFile{}-{}.demo"
        local_suffix = "local"
        object_suffix = "s3object"
        download_suffix = "downloaded"
        file_tag = len(self._chore_list) + 1

        local_file_path = os.path.join(
            self.demo_folder,
            file_name_template.format(file_tag, local_suffix))

        s3_object_key = file_name_template.format(file_tag, object_suffix)

        downloaded_file_path = os.path.join(
            self.demo_folder,
            file_name_template.format(file_tag, download_suffix))

        filled_cmd = self._create_file_cmd.format(
            local_file_path,
            self.file_size_mb * self._size_multiplier)

        print(f"Creating file of size {self.file_size_mb} MB "
              f"in {self.demo_folder} by running:")
        print(f"'{':4}{filled_cmd}'")
        os.system(filled_cmd)

        chore = (local_file_path, s3_object_key, downloaded_file_path)
        self._chore_list.append(chore)
        return chore

    def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
        """Report configuration and extra arguments used for a file transfer."""
        print('-' * self._terminal_width)
        print(f'{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}')
        if kwargs:
            print('With extra args:')
            for arg, value in kwargs.items():
```

```
        print(f'{"":4}{arg:<20}: {value}')
```

```
@staticmethod
def ask_user(question):
    """
    Ask the user a yes or no question.

    Returns:
    True when the user answers 'y' or 'Y'; otherwise, False.
    """
    answer = input(f'{question} (y/n) ')
    return answer.lower() == 'y'
```

```
@staticmethod
def _config_wrapper(func, config_attrs):
    def wrapper(*args, **kwargs):
        config = func(*args, **kwargs)
        print('With configuration:')
        for attr in config_attrs:
            print(f'{"":4}{attr:<20}: {getattr(config, attr)}')
        return config

    return wrapper
```

```
@staticmethod
def _report_transfer_result(thread_info, elapsed):
    """Report the result of a transfer, including per-thread data."""
    print(f"\nUsed {len(thread_info)} threads.")
    for ident, byte_count in thread_info.items():
        print(f'{"":4}Thread {ident} copied {byte_count} bytes.')
    print(f'Your transfer took {elapsed:.2f} seconds.')
```

```
def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file ",
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration)

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file ",
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold)

    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and ",
        "then download the same file using a single thread?",
        file_transfer.upload_with_chunksizes_and_meta,
        file_transfer.download_with_single_thread,
        upload_args={
```

```
'metadata': {
    'upload_type': 'chunky',
    'favorite_color': 'aqua',
    'size': 'medium'}})

# Upload using server-side encryption with customer-provided
# encryption keys.
# Generate a 256-bit key from a passphrase.
sse_key = hashlib.sha256('demo_passphrase'.encode('utf-8')).digest()
demo_manager.demo(
    "Do you want to upload and download a {} MB file using "
    "server-side encryption?",
    file_transfer.upload_with_sse,
    file_transfer.download_with_sse,
    upload_args={'sse_key': sse_key},
    download_args={'sse_key': sse_key})

# Download without specifying an encryption key to show that the
# encryption key must be included to download an encrypted object.
if demo_manager.ask_user("Do you want to try to download the encrypted "
                        "object without sending the required key?"):
    try:
        _, object_key, download_file_path = \
            demo_manager.last_name_set()
        file_transfer.download_with_default_configuration(
            demo_manager.demo_bucket, object_key, download_file_path,
            demo_manager.file_size_mb)
    except ClientError as err:
        print("Got expected error when trying to download an encrypted "
              "object without specifying encryption info:")
        print(f"'{err}'")

# Remove all created and downloaded files, remove all objects from
# S3 storage.
if demo_manager.ask_user(
        "Demonstration complete. Do you want to remove local files "
        "and S3 objects?"):
    demo_manager.cleanup()

if __name__ == '__main__':
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print("To run this example, you must have valid credentials in "
              "a shared credential file or set in environment variables.")
```

Rust

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
use std::convert::TryInto;
```

```
use std::fs::File;
use std::io::prelude::*;
use std::path::Path;

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::model::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::output::{CreateMultipartUploadOutput, GetObjectOutput};
use aws_sdk_s3::{Client as S3Client, Error, Region};
use aws_smithy_http::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::thread_rng::Rng;
use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() -> Result<(), Error> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("doc-example-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-west-2"));
    let region = region_provider.region().await.unwrap();
    s3_service::create_bucket(&client, &bucket_name, region.as_ref()).await?;

    let key = "sample.txt".to_string();
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await
        .unwrap();
    let upload_id = multipart_upload_res.upload_id().unwrap();

    //Create a file of random characters for the upload.
    let mut file = File::create(&key).expect("Could not create sample file.");
    // Loop until the file is 5 chunks.
    while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
        let rand_string: String = thread_rng()
            .sample_iter(&Alphanumeric)
            .take(256)
            .map(char::from)
            .collect();
        let return_string: String = "\n".to_string();
        file.write_all(rand_string.as_ref())
            .expect("Error writing to file.");
        file.write_all(return_string.as_ref())
            .expect("Error writing to file.");
    }

    let path = Path::new(&key);
    let file_size = tokio::fs::metadata(path)
        .await
        .expect("it exists I swear")
        .len();

    let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
    let mut size_of_last_chunk = file_size % CHUNK_SIZE;
    if size_of_last_chunk == 0 {
        size_of_last_chunk = CHUNK_SIZE;
        chunk_count -= 1;
    }
}
```

```

if file_size == 0 {
    panic!("Bad file size.");
}
if chunk_count > MAX_CHUNKS {
    panic!("Too many chunks! Try increasing your chunk size.")
}

let mut upload_parts: Vec<CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();
    //Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
        .await?;
    upload_parts.push(
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
            .part_number(part_number)
            .build(),
    );
}
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await
    .unwrap();

let data: GetObjectOutput = s3_service::download_object(&client, &bucket_name,
&key).await;
let data_length: u64 = data.content_length().try_into().unwrap();
if file.metadata().unwrap().len() == data_length {
    println!("Data lengths match.");
} else {
    println!("The data was not the same size!");
}

s3_service::delete_objects(&client, &bucket_name)
    .await

```

```
        .expect("Error emptying bucket.");
    s3_service::delete_bucket(&client, &bucket_name)
        .await
        .expect("Error deleting bucket.");

    Ok(())
}
```

## Work with Amazon S3 versioned objects using an AWS SDK

The following code example shows how to:

- Create a versioned S3 bucket.
- Get all versions of an object.
- Roll an object back to a previous version.
- Delete and restore a versioned object.
- Permanently delete all versions of an object.

### Python

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap S3 actions.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
    lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                'LocationConstraint': s3.meta.client.meta.region_name
            }
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response['Error']['Code'] == 'BucketAlreadyOwnedByYou':
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
        bucket = s3.Bucket(bucket_name)
```

```

        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                'Rules': [
                    {
                        'Status': 'Enabled',
                        'Prefix': prefix,
                        'NoncurrentVersionExpiration': {'NoncurrentDays': expiration}
                    }
                ]
            }
        )
        logger.info("Configured lifecycle to expire noncurrent versions after %s days"
days "
            "on bucket %s.", expiration, bucket.name)
    except ClientError as error:
        logger.warning("Couldn't configure lifecycle on bucket %s because %s. "
                      "Continuing anyway.", bucket.name, error)

    return bucket

def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are # at the end of the list even when they are interspersed in time.
    versions = sorted(bucket.object_versions.filter(Prefix=object_key),
                      key=attrgetter('last_modified'), reverse=True)

    logger.debug(
        "Got versions:\n%s",
        '\n'.join([f"\t{version.version_id}, last modified {version.last_modified}"
                  for version in versions]))

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(f"{version_id} was not found in the list of versions for "
                      f"{object_key}.")

```

```

def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
    marker.
    By removing the delete marker, we make the previous version the latest version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1)

    if 'DeleteMarkers' in response:
        latest_version = response['DeleteMarkers'][0]
        if latest_version['IsLatest']:
            logger.info("Object %s was indeed deleted on %s. Let's revive it.",
                        object_key, latest_version['LastModified'])
            obj = bucket.Object(object_key)
            obj.Version(latest_version['VersionId']).delete()
            logger.info("Revived %s, active version is now %s with body '%s'",
                        object_key, obj.version_id, obj.get()['Body'].read())
        else:
            logger.warning("Delete marker is not the latest version for %s!",
                           object_key)
    elif 'Versions' in response:
        logger.warning("Got an active version for %s, nothing to do.", object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)

def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise

```

Upload the stanza of a poem to a versioned object and perform a series of actions on it.

```

def usage_demo_single_object(obj_prefix='demo-versioning/'):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open('father_william.txt') as file:

```

```
    stanzas = file.read().split('\n\n')

    width = get_terminal_size((80, 20))[0]
    print('*'*width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
    print("This demonstration uploads a single stanza of a poem to an Amazon "
          "S3 bucket and then applies various revisions to it.")
    print('*'*width)
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket('bucket-' + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])

    # Add the first stanza and revise it a few times.
    print("\nApplying some revisions to the stanza...")
    obj_stanza_1 = bucket.Object(f'{obj_prefix}stanza-1')
    obj_stanza_1.put(Body=bytes(stanzas[0], 'utf-8'))
    obj_stanza_1.put(Body=bytes(stanzas[0].upper(), 'utf-8'))
    obj_stanza_1.put(Body=bytes(stanzas[0].lower(), 'utf-8'))
    obj_stanza_1.put(Body=bytes(stanzas[0][::-1], 'utf-8'))
    print("The latest version of the stanza is now:",
          obj_stanza_1.get()['Body'].read().decode('utf-8'),
          sep='\n')

    # Versions are returned in order, most recent first.
    obj_stanza_1_versions = bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    print(
        "The version data of the stanza revisions:",
        *[f" {version.version_id}, last modified {version.last_modified}"
          for version in obj_stanza_1_versions],
        sep='\n'
    )

    # Rollback two versions.
    print("\nRolling back two versions...")
    rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
    print("The latest version of the stanza:",
          obj_stanza_1.get()['Body'].read().decode('utf-8'),
          sep='\n')

    # Delete the stanza
    print("\nDeleting the stanza...")
    obj_stanza_1.delete()
    try:
        obj_stanza_1.get()
    except ClientError as error:
        if error.response['Error']['Code'] == 'NoSuchKey':
            print("The stanza is now deleted (as expected).")
        else:
            raise

    # Revive the stanza
    print("\nRestoring the stanza...")
    revive_object(bucket, obj_stanza_1.key)
    print("The stanza is restored! The latest version is again:",
          obj_stanza_1.get()['Body'].read().decode('utf-8'),
          sep='\n')

    # Permanently delete all versions of the object. This cannot be undone!
    print("\nPermanently deleting all versions of the stanza...")
    permanently_delete_object(bucket, obj_stanza_1.key)
    obj_stanza_1_versions = bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    if len(list(obj_stanza_1_versions)) == 0:
        print("The stanza has been permanently deleted and now has no versions.")
```

```
else:  
    print("Something went wrong. The stanza still exists!")  
  
print(f"\nRemoving {bucket.name}...")  
bucket.delete()  
print(f"{bucket.name} deleted.")  
print("Demo done!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateBucket](#)
  - [DeleteObject](#)
  - [ListObjectVersions](#)
  - [PutBucketLifecycleConfiguration](#)

## Cross-service examples for Amazon S3 using AWS SDKs

The following code examples show how to use Amazon Simple Storage Service with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Build an Amazon Transcribe app \(p. 1824\)](#)
- [Convert text to speech and back to text using an AWS SDK \(p. 1825\)](#)
- [Create a long-lived Amazon EMR cluster and run several steps using an AWS SDK \(p. 1825\)](#)
- [Create a short-lived Amazon EMR cluster and run a step using an AWS SDK \(p. 1826\)](#)
- [Create an Amazon Textract explorer application \(p. 1826\)](#)
- [Detect PPE in images with Amazon Rekognition using an AWS SDK \(p. 1827\)](#)
- [Detect entities in text extracted from an image using an AWS SDK \(p. 1828\)](#)
- [Detect faces in an image using an AWS SDK \(p. 1829\)](#)
- [Detect objects in images with Amazon Rekognition using an AWS SDK \(p. 1829\)](#)
- [Detect people and objects in a video with Amazon Rekognition using an AWS SDK \(p. 1832\)](#)
- [Save EXIF and other image information using an AWS SDK \(p. 1832\)](#)

## Build an Amazon Transcribe app

The following code example shows how to use Amazon Transcribe to transcribe and display voice recordings in the browser.

JavaScript

### SDK for JavaScript V3

Create an app that uses Amazon Transcribe to transcribe and display voice recordings in the browser. The app uses two Amazon Simple Storage Service (Amazon S3) buckets, one to host the application code, and another to store transcriptions. The app uses an Amazon Cognito user pool to authenticate your users. Authenticated users have AWS Identity and Access Management (IAM) permissions to access the required AWS services.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

### Services used in this example

- Amazon Cognito Identity
- Amazon S3
- Amazon Transcribe

## Convert text to speech and back to text using an AWS SDK

The following code example shows how to:

- Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file.
- Upload the audio file to an Amazon Simple Storage Service (Amazon S3) bucket.
- Use Amazon Transcribe to convert the audio file to text.
- Display the text.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file, upload the audio file to an Amazon Simple Storage Service bucket, use Amazon Transcribe to convert that audio file to text, and display the text.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Polly
- Amazon S3
- Amazon Transcribe

## Create a long-lived Amazon EMR cluster and run several steps using an AWS SDK

The following code example shows how to create a long-lived Amazon EMR cluster and run several steps.

Python

### SDK for Python (Boto3)

Create a long-lived Amazon EMR cluster that uses Apache Spark to query historical Amazon review data from the [Amazon Customer Reviews Dataset](#). Run a job that gets data for top-rated products in specific categories that contain keywords in their product titles. Job results are written to an Amazon Simple Storage Service (Amazon S3) bucket.

- Create an Amazon S3 bucket and upload a job script.
- Create AWS Identity and Access Management (IAM) roles.

- Create Amazon Elastic Compute Cloud (Amazon EC2) security groups.
- Create a long-lived cluster and run several job steps.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Amazon EC2
- Amazon EMR
- IAM
- Amazon S3

## Create a short-lived Amazon EMR cluster and run a step using an AWS SDK

The following code example shows how to create a short-lived Amazon EMR cluster that runs a step and automatically terminates after the step completes.

Python

#### **SDK for Python (Boto3)**

Create a short-lived Amazon EMR cluster that estimates the value of pi using Apache Spark to parallelize a large number of calculations. The job writes output to Amazon EMR logs and to an Amazon Simple Storage Service (Amazon S3) bucket. The cluster terminates itself after completing the job.

- Create an Amazon S3 bucket and upload a job script.
- Create AWS Identity and Access Management (IAM) roles.
- Create Amazon Elastic Compute Cloud (Amazon EC2) security groups.
- Create a short-lived cluster and run a single job step.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Amazon EC2
- Amazon EMR
- IAM
- Amazon S3

## Create an Amazon Textract explorer application

The following code examples show how to explore Amazon Textract output through an interactive application.

JavaScript

#### **SDK for JavaScript V3**

Shows how to use the AWS SDK for JavaScript to build a React application that uses Amazon Textract to extract data from a document image and display it in an interactive web page.

This example runs in a web browser and requires an authenticated Amazon Cognito identity for credentials. It uses Amazon Simple Storage Service (Amazon S3) for storage, and for notifications it polls an Amazon Simple Queue Service (Amazon SQS) queue that is subscribed to an Amazon Simple Notification Service (Amazon SNS) topic.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

#### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with Amazon Textract to detect text, form, and table elements in a document image. The input image and Amazon Textract output are shown in a Tkinter application that lets you explore the detected elements.

- Submit a document image to Amazon Textract and explore the output of detected elements.
- Submit images directly to Amazon Textract or through an Amazon Simple Storage Service (Amazon S3) bucket.
- Use asynchronous APIs to start a job that publishes a notification to an Amazon Simple Notification Service (Amazon SNS) topic when the job completes.
- Poll an Amazon Simple Queue Service (Amazon SQS) queue for a job completion message and display the results.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

## Detect PPE in images with Amazon Rekognition using an AWS SDK

The following code examples show how to build an app that uses Amazon Rekognition to detect Personal Protective Equipment (PPE) in images.

Java

#### SDK for Java 2.x

Shows how to create an AWS Lambda function that detects images with Personal Protective Equipment.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

#### SDK for JavaScript V3

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an application to detect personal protective equipment (PPE) in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app saves the results to an Amazon DynamoDB table, and sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for PPE using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Update a DynamoDB table with results.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

## Detect entities in text extracted from an image using an AWS SDK

The following code example shows how to use Amazon Comprehend to detect entities in text extracted by Amazon Textract from an image that is stored in Amazon S3.

Python

#### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) in a Jupyter notebook to detect entities in text that is extracted from an image. This example uses Amazon Textract to extract text from an image stored in Amazon Simple Storage Service (Amazon S3) and Amazon Comprehend to detect entities in the extracted text.

This example is a Jupyter notebook and must be run in an environment that can host notebooks. For instructions on how to run the example using Amazon SageMaker, see the directions in [TextractAndComprehendNotebook.ipynb](#).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon S3
- Amazon Textract

## Detect faces in an image using an AWS SDK

The following code example shows how to:

- Save an image in an Amazon Simple Storage Service Amazon S3 bucket.
- Use Amazon Rekognition (Amazon Rekognition) to detect facial details, such as age range, gender, and emotion (smiling, etc.).
- Display those details.

Rust

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

Save the image in an Amazon Simple Storage Service bucket with an **uploads** prefix, use Amazon Rekognition to detect facial details, such as age range, gender, and emotion (smiling, etc.), and display those details.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Rekognition
- Amazon S3

## Detect objects in images with Amazon Rekognition using an AWS SDK

The following code examples show how to build an app that uses Amazon Rekognition to detect objects by category in images.

.NET

#### AWS SDK for .NET

Shows how to use Amazon Rekognition .NET API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon

S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

#### **SDK for Java 2.x**

Shows how to use Amazon Rekognition Java API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

#### **SDK for JavaScript V3**

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for objects using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to use Amazon Rekognition Kotlin API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Python

### SDK for Python (Boto3)

Shows you how to use the AWS SDK for Python (Boto3) to create a web application that lets you do the following:

- Upload photos to an Amazon Simple Storage Service (Amazon S3) bucket.
- Use Amazon Rekognition to analyze and label the photos.
- Use Amazon Simple Email Service (Amazon SES) to send email reports of image analysis.

This example contains two main components: a webpage written in JavaScript that is built with React, and a REST service written in Python that is built with Flask-RESTful.

You can use the React webpage to:

- Display a list of images that are stored in your S3 bucket.
- Upload images from your computer to your S3 bucket.
- Display images and labels that identify items that are detected in the image.
- Get a report of all images in your S3 bucket and send an email of the report.

The webpage calls the REST service. The service sends requests to AWS to perform the following actions:

- Get and filter the list of images in your S3 bucket.
- Upload photos to your S3 bucket.
- Use Amazon Rekognition to analyze individual photos and get a list of labels that identify items that are detected in the photo.
- Analyze all photos in your S3 bucket and use Amazon SES to email a report.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Detect people and objects in a video with Amazon Rekognition using an AWS SDK

The following code examples show how to detect people and objects in a video with Amazon Rekognition.

Java

### SDK for Java 2.x

Shows how to use Amazon Rekognition Java API to create an app to detect faces and objects in videos located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

### SDK for JavaScript V3

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an app to detect faces and objects in videos located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for PPE using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Save EXIF and other image information using an AWS SDK

The following code example shows how to:

- Get EXIF information from a a JPG, JPEG, or PNG file.
- Upload the image file to an Amazon Simple Storage Service (Amazon S3) bucket.
- Use Amazon Rekognition (Amazon Rekognition) to identify the three top attributes (labels in Amazon Rekognition) in the file.

- Add the EXIF and label information to a Amazon DynamoDB (DynamoDB) table in the Region.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

Get EXIF information from a JPG, JPEG, or PNG file, upload the image file to an Amazon Simple Storage Service bucket, use Amazon Rekognition to identify the three top attributes (*labels* in Amazon Rekognition) in the file, and add the EXIF and label information to a Amazon DynamoDB table in the Region.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Amazon Rekognition
- Amazon S3

## Code examples for S3 Glacier using AWS SDKs

The following code examples show how to use Amazon S3 Glacier with an AWS software development kit (SDK).

### Code examples

- [Actions for S3 Glacier using AWS SDKs \(p. 1834\)](#)
  - Add tags to an Amazon S3 Glacier vault using an AWS SDK (p. 1834)
  - Create a multipart upload to an Amazon S3 Glacier vault using an AWS SDK (p. 1835)
  - Create an Amazon S3 Glacier vault using an AWS SDK (p. 1836)
  - Delete an Amazon S3 Glacier vault using an AWS SDK (p. 1839)
  - Delete an Amazon S3 Glacier archive using an AWS SDK (p. 1840)
  - Delete Amazon S3 Glacier vault notifications using an AWS SDK (p. 1841)
  - Describe an Amazon S3 Glacier job using an AWS SDK (p. 1842)
  - Download an Amazon S3 Glacier archive using an AWS SDK (p. 1843)
  - Get Amazon S3 Glacier job output using an AWS SDK (p. 1844)
  - Get Amazon S3 Glacier vault notification configuration using an AWS SDK (p. 1845)
  - List Amazon S3 Glacier jobs using an AWS SDK (p. 1846)
  - List tags for an Amazon S3 Glacier vault using an AWS SDK (p. 1848)
  - List Amazon S3 Glacier vaults using an AWS SDK (p. 1848)
  - Retrieve an Amazon S3 Glacier vault inventory using an AWS SDK (p. 1851)
  - Retrieve an archive from an Amazon S3 Glacier vault using an AWS SDK (p. 1853)
  - Set Amazon S3 Glacier vault notifications using an AWS SDK (p. 1854)
  - Upload an archive to an Amazon S3 Glacier vault using an AWS SDK (p. 1854)
- [Scenarios for S3 Glacier using AWS SDKs \(p. 1860\)](#)
  - Archive a file to Amazon S3 Glacier, get notifications, and initiate a job using an AWS SDK (p. 1860)

- Get Amazon S3 Glacier archive content and delete the archive using an AWS SDK (p. 1864)

## Actions for S3 Glacier using AWS SDKs

The following code examples show how to use Amazon S3 Glacier with AWS SDKs. Each example calls an individual service function.

### Examples

- Add tags to an Amazon S3 Glacier vault using an AWS SDK (p. 1834)
- Create a multipart upload to an Amazon S3 Glacier vault using an AWS SDK (p. 1835)
- Create an Amazon S3 Glacier vault using an AWS SDK (p. 1836)
- Delete an Amazon S3 Glacier vault using an AWS SDK (p. 1839)
- Delete an Amazon S3 Glacier archive using an AWS SDK (p. 1840)
- Delete Amazon S3 Glacier vault notifications using an AWS SDK (p. 1841)
- Describe an Amazon S3 Glacier job using an AWS SDK (p. 1842)
- Download an Amazon S3 Glacier archive using an AWS SDK (p. 1843)
- Get Amazon S3 Glacier job output using an AWS SDK (p. 1844)
- Get Amazon S3 Glacier vault notification configuration using an AWS SDK (p. 1845)
- List Amazon S3 Glacier jobs using an AWS SDK (p. 1846)
- List tags for an Amazon S3 Glacier vault using an AWS SDK (p. 1848)
- List Amazon S3 Glacier vaults using an AWS SDK (p. 1848)
- Retrieve an Amazon S3 Glacier vault inventory using an AWS SDK (p. 1851)
- Retrieve an archive from an Amazon S3 Glacier vault using an AWS SDK (p. 1853)
- Set Amazon S3 Glacier vault notifications using an AWS SDK (p. 1854)
- Upload an archive to an Amazon S3 Glacier vault using an AWS SDK (p. 1854)

## Add tags to an Amazon S3 Glacier vault using an AWS SDK

The following code example shows how to add tags to an Amazon S3 Glacier vault.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Glacier;
using Amazon.Glacier.Model;

public class AddTagsToVault
{
    public static async Task Main(string[] args)
    {
        string vaultName = "example-vault";

        var client = new AmazonGlacierClient();
        var request = new AddTagsToVaultRequest
```

```
        {
            Tags = new Dictionary<string, string>
            {
                { "examplekey1", "examplevalue1" },
                { "examplekey2", "examplevalue2" },
            },
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await client.AddTagsToVaultAsync(request);
    }
}
```

- For API details, see [AddTagsToVault](#) in *AWS SDK for .NET API Reference*.

## Create a multipart upload to an Amazon S3 Glacier vault using an AWS SDK

The following code example shows how to create a multipart upload to an Amazon S3 Glacier vault.

JavaScript

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a multipart upload of 1 megabyte chunks of a Buffer object.

```
// Create a new service object and some supporting variables
var glacier = new AWS.Glacier({apiVersion: '2012-06-01'}),
    vaultName = 'YOUR_VAULT_NAME',
    buffer = new Buffer(2.5 * 1024 * 1024), // 2.5MB buffer
    partSize = 1024 * 1024, // 1MB chunks,
    numPartsLeft = Math.ceil(buffer.length / partSize),
    startTime = new Date(),
    params = {vaultName: vaultName, partSize: partSize.toString()};

// Compute the complete SHA-256 tree hash so we can pass it
// to completeMultipartUpload request at the end
var treeHash = glacier.computeChecksums(buffer).treeHash;

// Initiate the multipart upload
console.log('Initiating upload to', vaultName);
// Call Glacier to initiate the upload.
glacier.initiateMultipartUpload(params, function (mpErr, multipart) {
    if (mpErr) { console.log('Error!', mpErr.stack); return; }
    console.log("Got upload ID", multipart.uploadId);

    // Grab each partSize chunk and upload it as a part
    for (var i = 0; i < buffer.length; i += partSize) {
        var end = Math.min(i + partSize, buffer.length),
            partParams = {
                vaultName: vaultName,
                uploadId: multipart.uploadId,
                range: 'bytes ' + i + '-' + (end-1) + '/*',
                body: buffer.slice(i, end)
            };
    }
});
```

```
// Send a single part
console.log('Uploading part', i, '=', partParams.range);
glacier.uploadMultipartPart(partParams, function(multiErr, mData) {
    if (multiErr) return;
    console.log("Completed part", this.request.params.range);
    if (--numPartsLeft > 0) return; // complete only when all parts
uploaded

    var doneParams = {
        vaultName: vaultName,
        uploadId: multipart.uploadId,
        archiveSize: buffer.length.toString(),
        checksum: treeHash // the computed tree hash
    };

    console.log("Completing upload...");
    glacier.completeMultipartUpload(doneParams, function(err, data) {
        if (err) {
            console.log("An error occurred while uploading the archive");
            console.log(err);
        } else {
            var delta = (new Date() - startTime) / 1000;
            console.log('Completed upload in', delta, 'seconds');
            console.log('Archive ID:', data.archiveId);
            console.log('Checksum: ', data.checksum);
        }
    });
});
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UploadMultipartPart](#) in [AWS SDK for JavaScript API Reference](#).

## Create an Amazon S3 Glacier vault using an AWS SDK

The following code examples show how to create an Amazon S3 Glacier vault.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Glacier;
using Amazon.Glacier.Model;

public class CreateVault
{
    static async Task Main(string[] args)
    {
        string vaultName = "example-vault";
        var client = new AmazonGlacierClient();
        var request = new CreateVaultRequest
        {
```

```
// Setting the AccountId to "-" means that
// the account associated with the default
// client will be used.
AccountId = "-",
VaultName = vaultName,
};

var response = await client.CreateVaultAsync(request);

Console.WriteLine($"Created {vaultName} at: {response.Location}");
}
```

- For API details, see [CreateVault in AWS SDK for .NET API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createGlacierVault(GlacierClient glacier, String vaultName ) {

    try {
        CreateVaultRequest vaultRequest = CreateVaultRequest.builder()
            .vaultName(vaultName)
            .build();

        CreateVaultResponse createVaultResult =
glacier.createVault(vaultRequest);
        System.out.println("The URI of the new vault is " +
createVaultResult.location());

    } catch(GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateVault in AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
```

```
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

Create the vault.

```
// Load the SDK for JavaScript
import { CreateVaultCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME
const params = { vaultName: vaultname };

const run = async () => {
  try {
    const data = await glacierClient.send(new CreateVaultCommand(params));
    console.log("Success, vault created!");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error");
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateVault in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the SDK for JavaScript
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create a new service object
var glacier = new AWS.Glacier({apiVersion: '2012-06-01'});
// Call Glacier to create the vault
glacier.createVault({vaultName: 'YOUR_VAULT_NAME'}, function(err) {
  if (!err) {
    console.log("Created vault!")
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateVault in AWS SDK for JavaScript API Reference](#).

## Python

### SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:  
    """Encapsulates Amazon S3 Glacier API operations."""  
    def __init__(self, glacier_resource):  
        """  
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.  
        """  
        self.glacier_resource = glacier_resource  
  
    def create_vault(self, vault_name):  
        """  
        Creates a vault.  
  
        :param vault_name: The name to give the vault.  
        :return: The newly created vault.  
        """  
        try:  
            vault = self.glacier_resource.create_vault(vaultName=vault_name)  
            logger.info("Created vault %s.", vault_name)  
        except ClientError:  
            logger.exception("Couldn't create vault %s.", vault_name)  
            raise  
        else:  
            return vault
```

- For API details, see [CreateVault in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete an Amazon S3 Glacier vault using an AWS SDK

The following code examples show how to delete an Amazon S3 Glacier vault.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteGlacierArchive(GlacierClient glacier, String  
vaultName, String accountId, String archiveId) {  
  
    try {  
        DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()  
            .vaultName(vaultName)  
            .accountId(accountId)  
            .archiveId(archiveId)  
            .build();  
  
        glacier.deleteArchive(delArcRequest);  
        System.out.println("The vault was deleted!");  
  
    } catch(GlacierException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteVault in AWS SDK for Java 2.x API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:  
    """Encapsulates Amazon S3 Glacier API operations."""  
    def __init__(self, glacier_resource):  
        """  
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.  
        """  
        self.glacier_resource = glacier_resource  
  
    @staticmethod  
    def delete_vault(vault):  
        """  
        Deletes a vault.  
  
        :param vault: The vault to delete.  
        """  
        try:  
            vault.delete()  
            logger.info("Deleted vault %s.", vault.name)  
        except ClientError:  
            logger.exception("Couldn't delete vault %s.", vault.name)  
            raise
```

- For API details, see [DeleteVault in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete an Amazon S3 Glacier archive using an AWS SDK

The following code examples show how to delete an Amazon S3 Glacier archive.

### Java

#### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteGlacierArchive(GlacierClient glacier, String  
vaultName, String accountId, String archiveId) {  
  
    try {  
        DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()  
            .vaultName(vaultName)  
            .accountId(accountId)  
            .archiveId(archiveId)  
            .build();  
  
        glacier.deleteArchive(delArcRequest);  
        System.out.println("The vault was deleted!");  
    } catch(GlacierException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());
```

```
        System.exit(1);
    }
}
```

- For API details, see [DeleteArchive](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    @staticmethod
    def delete_archive(archive):
        """
        Deletes an archive from a vault.

        :param archive: The archive to delete.
        """
        try:
            archive.delete()
            logger.info(
                "Deleted archive %s from vault %s.", archive.id,
                archive.vault_name)
        except ClientError:
            logger.exception("Couldn't delete archive %s.", archive.id)
            raise
```

- For API details, see [DeleteArchive](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete Amazon S3 Glacier vault notifications using an AWS SDK

The following code example shows how to delete Amazon S3 Glacier vault notifications.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
```

```
"""
    self.glacier_resource = glacier_resource

@staticmethod
def stop_notifications(notification):
    """
    Stops notifications to the configured Amazon SNS topic.

    :param notification: The notification configuration to remove.
    """
    try:
        notification.delete()
        logger.info("Notifications stopped.")
    except ClientError:
        logger.exception("Couldn't stop notifications.")
        raise
```

- For API details, see [DeleteVaultNotifications](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe an Amazon S3 Glacier job using an AWS SDK

The following code examples show how to describe an Amazon S3 Glacier job.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Glacier;
using Amazon.Glacier.Model;

public class DescribeVault
{
    public static async Task Main(string[] args)
    {
        string vaultName = "example-vault";
        var client = new AmazonGlacierClient();
        var request = new DescribeVaultRequest
        {
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await client.DescribeVaultAsync(request);

        // Display the information about the vault.
        Console.WriteLine($"{response.VaultName}\tARN: {response.VaultARN}");
        Console.WriteLine($"Created on: {response.CreationDate}\tNumber of
Archives: {response.NumberOfArchives}\tSize (in bytes): {response.SizeInBytes}");
        if (response.LastInventoryDate != DateTime.MinValue)
        {
            Console.WriteLine($"Last inventory: {response.LastInventoryDate}");
        }
    }
}
```

- For API details, see [DescribeJob in AWS SDK for .NET API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:  
    """Encapsulates Amazon S3 Glacier API operations."""  
    def __init__(self, glacier_resource):  
        """  
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.  
        """  
        self.glacier_resource = glacier_resource  
  
    @staticmethod  
    def get_job_status(job):  
        """  
        Gets the status of a job.  
  
        :param job: The job to query.  
        :return: The current status of the job.  
        """  
        try:  
            job.load()  
            logger.info(  
                "Job %s is performing action %s and has status %s.", job.id,  
                job.action, job.status_code)  
        except ClientError:  
            logger.exception("Couldn't get status for job %s.", job.id)  
            raise  
        else:  
            return job.status_code
```

- For API details, see [DescribeJob in AWS SDK for Python \(Boto3\) API Reference](#).

## Download an Amazon S3 Glacier archive using an AWS SDK

The following code example shows how to download an Amazon S3 Glacier archive.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using Amazon;  
using Amazon.Glacier;  
using Amazon.Glacier.Transfer;
```

```
using Amazon.Runtime;

class DownloadArchiveHighLevel
{
    private static readonly string VaultName = "examplevault";
    private static readonly string ArchiveId = "*** Provide archive ID ***";
    private static readonly string DownloadFilePath = "*** Provide the file
name and path to where to store the download ***";
    private static int currentPercentage = -1;

    static void Main()
    {
        try
        {
            var manager = new ArchiveTransferManager(RegionEndpoint.USEast2);

            var options = new DownloadOptions
            {
                StreamTransferProgress = Progress,
            };

            // Download an archive.
            Console.WriteLine("Initiating the archive retrieval job and then
polling SQS queue for the archive to be available.");
            Console.WriteLine("Once the archive is available, downloading will
begin.");
            manager.DownloadAsync(VaultName, ArchiveId, DownloadFilePath,
options);
            Console.WriteLine("To continue, press Enter");
            Console.ReadKey();
        }
        catch (AmazonGlacierException ex)
        {
            Console.WriteLine(ex.Message);
        }

        Console.WriteLine("To continue, press Enter");
        Console.ReadKey();
    }

    static void Progress(object sender, StreamTransferProgressArgs args)
    {
        if (args.PercentDone != currentPercentage)
        {
            currentPercentage = args.PercentDone;
            Console.WriteLine("Downloaded {0}%", args.PercentDone);
        }
    }
}
```

## Get Amazon S3 Glacier job output using an AWS SDK

The following code example shows how to get Amazon S3 Glacier job output.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:  
    """Encapsulates Amazon S3 Glacier API operations."""  
    def __init__(self, glacier_resource):  
        """  
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.  
        """  
        self.glacier_resource = glacier_resource  
  
    @staticmethod  
    def get_job_output(job):  
        """  
        Gets the output of a job, such as a vault inventory or the contents of an  
        archive.  
  
        :param job: The job to get output from.  
        :return: The job output, in bytes.  
        """  
        try:  
            response = job.get_output()  
            out_bytes = response['body'].read()  
            logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)  
            if 'archiveDescription' in response:  
                logger.info(  
                    "These bytes are described as '%s'",  
                    response['archiveDescription'])  
            except ClientError:  
                logger.exception("Couldn't get output for job %s.", job.id)  
                raise  
            else:  
                return out_bytes
```

- For API details, see [GetJobOutput](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get Amazon S3 Glacier vault notification configuration using an AWS SDK

The following code example shows how to get Amazon S3 Glacier vault notification configuration.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:  
    """Encapsulates Amazon S3 Glacier API operations."""  
    def __init__(self, glacier_resource):  
        """  
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.  
        """  
        self.glacier_resource = glacier_resource  
  
    @staticmethod  
    def get_notification(vault):
```

```
"""
Gets the currently notification configuration for a vault.

:param vault: The vault to query.
:return: The notification configuration for the specified vault.
"""
try:
    notification = vault.Notification()
    logger.info(
        "Vault %s notifies %s on %s events.", vault.name,
        notification.sns_topic, notification.events)
except ClientError:
    logger.exception("Couldn't get notification data for %s.", vault.name)
    raise
else:
    return notification
```

- For API details, see [GetVaultNotifications in AWS SDK for Python \(Boto3\) API Reference](#).

## List Amazon S3 Glacier jobs using an AWS SDK

The following code examples show how to list Amazon S3 Glacier jobs.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Glacier;
using Amazon.Glacier.Model;

class ListJobs
{
    static async Task Main(string[] args)
    {
        var client = new AmazonGlacierClient();
        var vaultName = "example-vault";

        var request = new ListJobsRequest
        {
            // Using a hyphen "=" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the default user.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await client.ListJobsAsync(request);

        if (response.JobList.Count > 0)
        {
            response.JobList.ForEach(job => {
                Console.WriteLine($"{job.CreationDate} {job.JobDescription}");
            });
        }
    }
}
```

```
        }
    else
    {
        Console.WriteLine($"No jobs were found for {vaultName}.");
    }
}
```

- For API details, see [ListJobs](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == 'all':
                jobs = vault.jobs.all()
            elif job_type == 'in_progress':
                jobs = vault.jobs_in_progress.all()
            elif job_type == 'completed':
                jobs = vault.completed_jobs.all()
            elif job_type == 'succeeded':
                jobs = vault.succeeded_jobs.all()
            elif job_type == 'failed':
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
                logger.warning("%s isn't a type of job I can get.", job_type)
            for job in jobs:
                job_list.append(job)
                logger.info("Got %s %s job %s.", job_type, job.action, job.id)
        except ClientError:
            logger.exception("Couldn't get %s jobs from %s.", job_type, vault.name)
            raise
        else:
            return job_list
```

- For API details, see [ListJobs](#) in *AWS SDK for Python (Boto3) API Reference*.

## List tags for an Amazon S3 Glacier vault using an AWS SDK

The following code example shows how to list tags for an Amazon S3 Glacier vault.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Glacier;
using Amazon.Glacier.Model;

class ListTagsForVault
{
    static async Task Main(string[] args)
    {
        var client = new AmazonGlacierClient();
        var vaultName = "example-vault";

        var request = new ListTagsForVaultRequest
        {
            // Using a hyphen "=" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the default user.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await client.ListTagsForVaultAsync(request);

        if (response.Tags.Count > 0)
        {
            foreach (KeyValuePair<string, string> tag in response.Tags)
            {
                Console.WriteLine($"Key: {tag.Key}, value: {tag.Value}");
            }
        }
        else
        {
            Console.WriteLine($"{vaultName} has no tags.");
        }
    }
}
```

- For API details, see [ListTagsForVault](#) in *AWS SDK for .NET API Reference*.

## List Amazon S3 Glacier vaults using an AWS SDK

The following code examples show how to list Amazon S3 Glacier vaults.

.NET

**AWS SDK for .NET**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Glacier;
using Amazon.Glacier.Model;

public class ListVaults
{
    public static async Task Main(string[] args)
    {
        var client = new AmazonGlacierClient();
        var request = new ListVaultsRequest
        {
            AccountId = "-",
            Limit = 5,
        };

        var response = await client.ListVaultsAsync(request);

        List<DescribeVaultOutput> vaultList = response.VaultList;

        vaultList.ForEach(v => { Console.WriteLine($"{v.VaultName} ARN: {v.VaultARN}"); });
    }
}
```

- For API details, see [ListVaults](#) in *AWS SDK for .NET API Reference*.

Java

**SDK for Java 2.x**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllVault(GlacierClient glacier) {

    boolean listComplete = false;
    String newMarker = null;
    int totalVaults = 0;
    System.out.println("Your Amazon Glacier vaults:");

    try {

        while (!listComplete) {
            ListVaultsResponse response = null;
            if (newMarker != null) {
                ListVaultsRequest request = ListVaultsRequest.builder()
                    .marker(newMarker)
```

```
        .build();

        response = glacier.listVaults(request);
    } else {
        ListVaultsRequest request = ListVaultsRequest.builder()
            .build();
        response = glacier.listVaults(request);
    }

    List<DescribeVaultOutput> vaultList = response.vaultList();
    for (DescribeVaultOutput v: vaultList) {
        totalVaults += 1;
        System.out.println("* " + v.vaultName());
    }

    // Check for further results.
    newMarker = response.marker();
    if (newMarker == null) {
        listComplete = true;
    }
}

if (totalVaults == 0) {
    System.out.println("No vaults found.");
}

} catch(GlacierException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [ListVaults](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    def list_vaults(self):
        """
        Lists vaults for the current account.
        """
        try:
            for vault in self.glacier_resource.vaults.all():
                logger.info("Got vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise
```

- For API details, see [ListVaults](#) in *AWS SDK for Python (Boto3) API Reference*.

## Retrieve an Amazon S3 Glacier vault inventory using an AWS SDK

The following code examples show how to retrieve an Amazon S3 Glacier vault inventory.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createJob(GlacierClient glacier, String vaultName, String
accountID) {

    try {

        JobParameters job = JobParameters.builder()
            .type("inventory-retrieval")
            .build();

        InitiateJobRequest initJob = InitiateJobRequest.builder()
            .jobParameters(job)
            .accountId(accountID)
            .vaultName(vaultName)
            .build();

        InitiateJobResponse response = glacier.initiateJob(initJob);
        System.out.println("The job ID is: " +response.jobId());
        System.out.println("The relative URI path of the job is: "
+response.location());
        return response.jobId();

    } catch(GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);

    }
    return "";
}

// Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
Documentation.
public static void checkJob(GlacierClient glacier, String jobId, String name,
String account, String path) {

    try{
        boolean finished = false;
        String jobStatus;
        int yy=0;

        while (!finished) {
            DescribeJobRequest jobRequest = DescribeJobRequest.builder()
                .jobId(jobId)
                .accountId(account)
                .vaultName(name)
```

```
.build();

DescribeJobResponse response = glacier.describeJob(jobRequest);
jobStatus = response.statusCodeAsString();

if (jobStatus.compareTo("Succeeded") == 0)
    finished = true;
else {
    System.out.println(yy + " status is: " + jobStatus);
    Thread.sleep(1000);
}
yy++;
}

System.out.println("Job has Succeeded");
GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
    . jobId(jobId)
    . vaultName(name)
    . accountId(account)
    . build();

ResponseBytes<GetJobOutputResponse> objectBytes =
glacier.getJobOutputAsBytes(jobOutputRequest);
// Write the data to a local file.
byte[] data = objectBytes.asByteArray();
File myFile = new File(path);
OutputStream os = new FileOutputStream(myFile);
os.write(data);
System.out.println("Successfully obtained bytes from a Glacier vault");
os.close();

} catch(GlacierException | InterruptedException | IOException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [InitiateJob](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    @staticmethod
    def initiate_inventory_retrieval(vault):
        """
        Initiates an inventory retrieval job. The inventory describes the contents
        of the vault. Standard retrievals typically complete within 3-5 hours.

```

When the job completes, you can get the inventory by calling `get_output()`.

```
:param vault: The vault to inventory.  
:return: The inventory retrieval job.  
"""  
try:  
    job = vault.initiate_inventory_retrieval()  
    logger.info("Started %s job with ID %s.", job.action, job.id)  
except ClientError:  
    logger.exception("Couldn't start job on vault %s.", vault.name)  
    raise  
else:  
    return job
```

- For API details, see [InitiateJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Retrieve an archive from an Amazon S3 Glacier vault using an AWS SDK

The following code example shows how to retrieve an archive from an Amazon S3 Glacier vault.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:  
    """Encapsulates Amazon S3 Glacier API operations."""  
    def __init__(self, glacier_resource):  
        """  
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.  
        """  
        self.glacier_resource = glacier_resource  
  
    @staticmethod  
    def initiate_archive_retrieval(archive):  
        """  
        Initiates an archive retrieval job. Standard retrievals typically complete  
        within 3–5 hours. When the job completes, you can get the archive contents  
        by calling get_output().  
        :param archive: The archive to retrieve.  
        :return: The archive retrieval job.  
        """  
        try:  
            job = archive.initiate_archive_retrieval()  
            logger.info("Started %s job with ID %s.", job.action, job.id)  
        except ClientError:  
            logger.exception("Couldn't start job on archive %s.", archive.id)  
            raise  
        else:  
            return job
```

- For API details, see [InitiateJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Set Amazon S3 Glacier vault notifications using an AWS SDK

The following code example shows how to set Amazon S3 Glacier vault notifications.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:  
    """Encapsulates Amazon S3 Glacier API operations."""  
    def __init__(self, glacier_resource):  
        """  
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.  
        """  
        self.glacier_resource = glacier_resource  
  
    def set_notifications(self, vault, sns_topic_arn):  
        """  
        Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target  
        for notifications. Amazon S3 Glacier publishes messages to this topic for  
        the configured list of events.  
  
        :param vault: The vault to set up to publish notifications.  
        :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that  
            receives notifications.  
        :return: Data about the new notification configuration.  
        """  
        try:  
            notification = self.glacier_resource.Notification('-', vault.name)  
            notification.set(vaultNotificationConfig={  
                'SNSTopic': sns_topic_arn,  
                'Events': ['ArchiveRetrievalCompleted',  
                          'InventoryRetrievalCompleted']  
            })  
            logger.info(  
                "Notifications will be sent to %s for events %s from %s.",  
                notification.sns_topic, notification.events,  
                notification.vault_name)  
        except ClientError:  
            logger.exception(  
                "Couldn't set notifications to %s on %s.", sns_topic_arn,  
                vault.name)  
            raise  
        else:  
            return notification
```

- For API details, see [SetVaultNotifications in AWS SDK for Python \(Boto3\) API Reference](#).

## Upload an archive to an Amazon S3 Glacier vault using an AWS SDK

The following code examples show how to upload an archive to an Amazon S3 Glacier vault.

.NET

**AWS SDK for .NET**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;

public class UploadArchiveHighLevel
{
    private static readonly string VaultName = "example-vault";
    private static readonly string ArchiveToUpload = "*** Provide file name
(with full path) to upload ***";

    public static async Task Main()
    {
        try
        {
            var manager = new ArchiveTransferManager(RegionEndpoint.USWest2);

            // Upload an archive.
            var response = await manager.UploadAsync(VaultName, "upload archive
test", ArchiveToUpload);

            Console.WriteLine("Copy and save the ID for use in other
examples.");
            Console.WriteLine($"Archive ID: {response.ArchiveId}");
            Console.WriteLine("To continue, press Enter");
            Console.ReadKey();
        }
        catch (AmazonGlacierException ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}
```

- For API details, see [UploadArchive](#) in *AWS SDK for .NET API Reference*.

Java

**SDK for Java 2.x**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String uploadContent(GlacierClient glacier, Path path, String
vaultName, File myFile) {

    // Get an SHA-256 tree hash value.
    String checkVal = computeSHA256(myFile);
    try {
```

```
UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
    .vaultName(vaultName)
    .checksum(checkVal)
    .build();

    UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
    return res.archiveId();

} catch(GlacierException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

private static String computeSHA256(File inputFile) {

    try {
        byte[] treeHash = computeSHA256TreeHash(inputFile);
        System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
        return toHex(treeHash);

    } catch (IOException ioe) {
        System.err.format("Exception when reading from file %s: %s", inputFile,
ioe.getMessage());
        System.exit(-1);

    } catch (NoSuchAlgorithmException nsae) {
        System.err.format("Cannot locate MessageDigest algorithm for SHA-256:
%s", nsae.getMessage());
        System.exit(-1);
    }
    return "";
}

public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
NoSuchAlgorithmException {

    byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
    return computeSHA256TreeHash(chunkSHA256Hashes);
}

/**
 * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
 * includes the checksum for the last chunk, even if it's smaller than 1 MB.
 */
public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    long numChunks = file.length() / ONE_MB;
    if (file.length() % ONE_MB > 0) {
        numChunks++;
    }

    if (numChunks == 0) {
        return new byte[][] { md.digest() };
    }

    byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
    FileInputStream fileStream = null;

    try {
        fileStream = new FileInputStream(file);
        byte[] buff = new byte[ONE_MB];
```

```

        int bytesRead;
        int idx = 0;

        while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
            md.reset();
            md.update(buff, 0, bytesRead);
            chunkSHA256Hashes[idx++] = md.digest();
        }

        return chunkSHA256Hashes;
    } finally {
        if (fileStream != null) {
            try {
                fileStream.close();
            } catch (IOException ioe) {
                System.err.printf("Exception while closing %s.\n %s",
file.getName(),
                        ioe.getMessage());
            }
        }
    }
}

/**
 * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
 * checksums.
 */
public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
throws NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    byte[][] prevLvlHashes = chunkSHA256Hashes;
    while (prevLvlHashes.length > 1) {
        int len = prevLvlHashes.length / 2;
        if (prevLvlHashes.length % 2 != 0) {
            len++;
        }

        byte[][] currLvlHashes = new byte[len][];
        int j = 0;
        for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {

            // If there are at least two elements remaining.
            if (prevLvlHashes.length - i > 1) {

                // Calculate a digest of the concatenated nodes.
                md.reset();
                md.update(prevLvlHashes[i]);
                md.update(prevLvlHashes[i + 1]);
                currLvlHashes[j] = md.digest();

            } else { // Take care of the remaining odd chunk
                currLvlHashes[j] = prevLvlHashes[i];
            }
        }

        prevLvlHashes = currLvlHashes;
    }

    return prevLvlHashes[0];
}

/**
 * Returns the hexadecimal representation of the input byte array
*/

```

```
public static String toHex(byte[] data) {
    StringBuilder sb = new StringBuilder(data.length * 2);
    for (byte datum : data) {
        String hex = Integer.toHexString(datum & 0xFF);

        if (hex.length() == 1) {
            // Append leading zero.
            sb.append("0");
        }
        sb.append(hex);
    }
    return sb.toString().toLowerCase();
}
```

- For API details, see [UploadArchive](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

Upload the archive.

```
// Load the SDK for JavaScript
import { UploadArchiveCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME

// Create a new service object and buffer
const buffer = new Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer
const params = { vaultName: vaultname, body: buffer };

const run = async () => {
    try {
        const data = await glacierClient.send(new UploadArchiveCommand(params));
        console.log("Archive ID", data.archiveId);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error uploading archive!", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [UploadArchive](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the SDK for JavaScript
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create a new service object and buffer
var glacier = new AWS.Glacier({apiVersion: '2012-06-01'});
buffer = Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer

var params = {vaultName: 'YOUR_VAULT_NAME', body: buffer};
// Call Glacier to upload the archive.
glacier.uploadArchive(params, function(err, data) {
  if (err) {
    console.log("Error uploading archive!", err);
  } else {
    console.log("Archive ID", data.archiveId);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UploadArchive](#) in *AWS SDK for JavaScript API Reference*.

## Python

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    @staticmethod
    def upload_archive(vault, archive_description, archive_file):
        """
        Uploads an archive to a vault.

        :param vault: The vault where the archive is put.
        :param archive_description: A description of the archive.
        :param archive_file: The archive file to put in the vault.
        :return: The uploaded archive.
        """
        try:
            archive = vault.upload_archive(
```

```
        archiveDescription=archive_description, body=archive_file)
    logger.info(
        "Uploaded %s with ID %s to vault %s.", archive_description,
        archive.id, vault.name)
except ClientError:
    logger.exception(
        "Couldn't upload %s to %s.", archive_description, vault.name)
    raise
else:
    return archive
```

- For API details, see [UploadArchive](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios for S3 Glacier using AWS SDKs

The following code examples show how to use Amazon S3 Glacier with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Archive a file to Amazon S3 Glacier, get notifications, and initiate a job using an AWS SDK \(p. 1860\)](#)
- [Get Amazon S3 Glacier archive content and delete the archive using an AWS SDK \(p. 1864\)](#)

## Archive a file to Amazon S3 Glacier, get notifications, and initiate a job using an AWS SDK

The following code example shows how to:

- Create an Amazon S3 Glacier vault.
- Configure the vault to publish notifications to an Amazon Simple Notification Service (Amazon SNS) topic.
- Upload an archive file to the vault.
- Initiate an archive retrieval job.

### Python

#### [SDK for Python \(Boto3\)](#)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps S3 Glacier operations.

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class GlacierWrapper:
```

```
"""Encapsulates Amazon S3 Glacier API operations."""
def __init__(self, glacier_resource):
    """
    :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
    """
    self.glacier_resource = glacier_resource

def create_vault(self, vault_name):
    """
    Creates a vault.

    :param vault_name: The name to give the vault.
    :return: The newly created vault.
    """
    try:
        vault = self.glacier_resource.create_vault(vaultName=vault_name)
        logger.info("Created vault %s.", vault_name)
    except ClientError:
        logger.exception("Couldn't create vault %s.", vault_name)
        raise
    else:
        return vault

def list_vaults(self):
    """
    Lists vaults for the current account.

    """
    try:
        for vault in self.glacier_resource.vaults.all():
            logger.info("Got vault %s.", vault.name)
    except ClientError:
        logger.exception("Couldn't list vaults.")
        raise

@staticmethod
def upload_archive(vault, archive_description, archive_file):
    """
    Uploads an archive to a vault.

    :param vault: The vault where the archive is put.
    :param archive_description: A description of the archive.
    :param archive_file: The archive file to put in the vault.
    :return: The uploaded archive.
    """
    try:
        archive = vault.upload_archive(
            archiveDescription=archive_description, body=archive_file)
        logger.info(
            "Uploaded %s with ID %s to vault %s.", archive_description,
            archive.id, vault.name)
    except ClientError:
        logger.exception(
            "Couldn't upload %s to %s.", archive_description, vault.name)
        raise
    else:
        return archive

@staticmethod
def initiate_archive_retrieval(archive):
    """
    Initiates an archive retrieval job. Standard retrievals typically complete
    within 3-5 hours. When the job completes, you can get the archive contents
    by calling get_output().

    :param archive: The archive to retrieve.
```

```
:return: The archive retrieval job.  
"""  
try:  
    job = archive.initiate_archive_retrieval()  
    logger.info("Started %s job with ID %s.", job.action, job.id)  
except ClientError:  
    logger.exception("Couldn't start job on archive %s.", archive.id)  
    raise  
else:  
    return job  
  
@staticmethod  
def list_jobs(vault, job_type):  
    """  
    Lists jobs by type for the specified vault.  
  
    :param vault: The vault to query.  
    :param job_type: The type of job to list.  
    :return: The list of jobs of the requested type.  
    """  
    job_list = []  
    try:  
        if job_type == 'all':  
            jobs = vault.jobs.all()  
        elif job_type == 'in_progress':  
            jobs = vault.jobs_in_progress.all()  
        elif job_type == 'completed':  
            jobs = vault.completed_jobs.all()  
        elif job_type == 'succeeded':  
            jobs = vault.succeeded_jobs.all()  
        elif job_type == 'failed':  
            jobs = vault.failed_jobs.all()  
        else:  
            jobs = []  
            logger.warning("%s isn't a type of job I can get.", job_type)  
        for job in jobs:  
            job_list.append(job)  
            logger.info("Got %s %s job %s.", job_type, job.action, job.id)  
    except ClientError:  
        logger.exception("Couldn't get %s jobs from %s.", job_type, vault.name)  
        raise  
    else:  
        return job_list  
  
def set_notifications(self, vault, sns_topic_arn):  
    """  
    Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target  
    for notifications. Amazon S3 Glacier publishes messages to this topic for  
    the configured list of events.  
  
    :param vault: The vault to set up to publish notifications.  
    :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that  
                        receives notifications.  
    :return: Data about the new notification configuration.  
    """  
    try:  
        notification = self.glacier_resource.Notification('-', vault.name)  
        notification.set(vaultNotificationConfig={  
            'SNSTopic': sns_topic_arn,  
            'Events': ['ArchiveRetrievalCompleted',  
                      'InventoryRetrievalCompleted']  
        })  
        logger.info(  
            "Notifications will be sent to %s for events %s from %s.",  
            notification.sns_topic, notification.events,  
            notification.vault_name)
```

```
        except ClientError:
            logger.exception(
                "Couldn't set notifications to %s on %s.", sns_topic_arn,
            vault.name)
            raise
        else:
            return notification
```

Call functions on the wrapper class to create a vault and upload a file, then configure the vault to publish notifications and initiate a job to retrieve the archive.

```
def upload_demo(glacier, vault_name, topic_arn):
    """
    Shows how to:
    * Create a vault.
    * Configure the vault to publish notifications to an Amazon SNS topic.
    * Upload an archive.
    * Start a job to retrieve the archive.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to create.
    :param topic_arn: The ARN of an Amazon SNS topic that receives notification of
                      Amazon S3 Glacier events.
    """
    print(f"\nCreating vault {vault_name}.")
    vault = glacier.create_vault(vault_name)
    print("\nList of vaults in your account:")
    glacier.list_vaults()
    print(f"\nUploading glacier_basics.py to {vault.name}.")
    with open("glacier_basics.py", 'rb') as upload_file:
        archive = glacier.upload_archive(vault, "glacier_basics.py", upload_file)
    print("\nStarting an archive retrieval request to get the file back from the "
         "vault.")
    glacier.initiate_archive_retrieval(archive)
    print("\nListing in progress jobs:")
    glacier.list_jobs(vault, 'in_progress')
    print("\nBecause Amazon S3 Glacier is intended for infrequent retrieval, an "
         "archive request with Standard retrieval typically completes within 3-5 "
         "hours.")
    if topic_arn:
        notification = glacier.set_notifications(vault, topic_arn)
        print(f"\nVault {vault.name} is configured to notify the "
              f"{notification.sns_topic} topic when {notification.events} "
              f"events occur. You can subscribe to this topic to receive "
              f"a message when the archive retrieval completes.\n")
    else:
        print(f"\nVault {vault.name} is not configured to notify an Amazon SNS "
              "topic "
              f"when the archive retrieval completes so wait a few hours.")
    print("\nRetrieve your job output by running this script with the --retrieve "
          "flag.")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateVault](#)
  - [InitiateJob](#)
  - [ListJobs](#)
  - [ListVaults](#)
  - [SetVaultNotifications](#)
  - [UploadArchive](#)

## Get Amazon S3 Glacier archive content and delete the archive using an AWS SDK

The following code example shows how to:

- List jobs for an Amazon S3 Glacier vault and get job status.
- Get the output of a completed archive retrieval job.
- Delete an archive.
- Delete a vault.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps S3 Glacier operations.

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
            if job_type == 'all':
                jobs = vault.jobs.all()
            elif job_type == 'in_progress':
                jobs = vault.jobs_in_progress.all()
            elif job_type == 'completed':
                jobs = vault.completed_jobs.all()
            elif job_type == 'succeeded':
                jobs = vault.succeeded_jobs.all()
            elif job_type == 'failed':
                jobs = vault.failed_jobs.all()
            else:
                jobs = []
        except ClientError as e:
            logger.error(f'Failed to list jobs: {e}')
            return None
        return jobs
```

```
        logger.warning("%s isn't a type of job I can get.", job_type)
    for job in jobs:
        job_list.append(job)
        logger.info("Got %s %s job %s.", job_type, job.action, job.id)
    except ClientError:
        logger.exception("Couldn't get %s jobs from %s.", job_type, vault.name)
        raise
    else:
        return job_list

@staticmethod
def get_job_output(job):
    """
    Gets the output of a job, such as a vault inventory or the contents of an
    archive.

    :param job: The job to get output from.
    :return: The job output, in bytes.
    """
    try:
        response = job.get_output()
        out_bytes = response['body'].read()
        logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)
        if 'archiveDescription' in response:
            logger.info(
                "These bytes are described as '%s',",
                response['archiveDescription'])
    except ClientError:
        logger.exception("Couldn't get output for job %s.", job.id)
        raise
    else:
        return out_bytes

@staticmethod
def delete_archive(archive):
    """
    Deletes an archive from a vault.

    :param archive: The archive to delete.
    """
    try:
        archive.delete()
        logger.info(
            "Deleted archive %s from vault %s.", archive.id,
            archive.vault_name)
    except ClientError:
        logger.exception("Couldn't delete archive %s.", archive.id)
        raise

@staticmethod
def delete_vault(vault):
    """
    Deletes a vault.

    :param vault: The vault to delete.
    """
    try:
        vault.delete()
        logger.info("Deleted vault %s.", vault.name)
    except ClientError:
        logger.exception("Couldn't delete vault %s.", vault.name)
        raise
```

Call functions on the wrapper class to get archive content from a completed job, then delete the archive.

```
def retrieve_demo(glacier, vault_name):
    """
    Shows how to:
    * List jobs for a vault and get job status.
    * Get the output of a completed archive retrieval job.
    * Delete an archive.
    * Delete a vault.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to query for jobs.
    """
    vault = glacier.glacier_resource.Vault('-', vault_name)
    try:
        vault.load()
    except ClientError as err:
        if err.response['Error']['Code'] == 'ResourceNotFoundException':
            print(f"\nVault {vault_name} doesn't exist. You must first run this
script "
                  f"with the --upload flag to create the vault.")
            return
        else:
            raise

    print(f"\nGetting completed jobs for {vault.name}.")
    jobs = glacier.list_jobs(vault, 'completed')
    if not jobs:
        print("\nNo completed jobs found. Give it some time and try again later.")
        return

    retrieval_job = None
    for job in jobs:
        if job.action == 'ArchiveRetrieval' and job.status_code == 'Succeeded':
            retrieval_job = job
            break
    if retrieval_job is None:
        print("\nNo ArchiveRetrieval jobs found. Give it some time and try again "
              "later.")
        return

    print(f"\nGetting output from job {retrieval_job.id}.")
    archive_bytes = glacier.get_job_output(retrieval_job)
    archive_str = archive_bytes.decode('utf-8')
    print("\nGot archive data. Printing the first 10 lines.")
    print(os.linesep.join(archive_str.split(os.linesep)[:10]))

    print(f"\nDeleting the archive from {vault.name}.")
    archive = glacier.glacier_resource.Archive(
        '-', vault.name, retrieval_job.archive_id)
    glacier.delete_archive(archive)

    print(f"\nDeleting {vault.name}.")
    glacier.delete_vault(vault)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [DeleteArchive](#)
  - [DeleteVault](#)
  - [GetJobOutput](#)
  - [ListJobs](#)

# Code examples for Amazon SES using AWS SDKs

The following code examples show how to use Amazon Simple Email Service with an AWS software development kit (SDK).

## Code examples

- [Actions for Amazon SES using AWS SDKs \(p. 1868\)](#)
  - Create an Amazon SES receipt filter using an AWS SDK (p. 1868)
  - Create an Amazon SES receipt rule using an AWS SDK (p. 1869)
  - Create an Amazon SES receipt rule set using an AWS SDK (p. 1871)
  - Create an Amazon SES email template using an AWS SDK (p. 1871)
  - Delete an Amazon SES receipt filter using an AWS SDK (p. 1873)
  - Delete an Amazon SES receipt rule using an AWS SDK (p. 1874)
  - Delete an Amazon SES rule set using an AWS SDK (p. 1874)
  - Delete an Amazon SES email template using an AWS SDK (p. 1875)
  - Delete an Amazon SES identity using an AWS SDK (p. 1877)
  - Describe an Amazon SES receipt rule set using an AWS SDK (p. 1878)
  - Get an existing Amazon SES email template using an AWS SDK (p. 1879)
  - Get Amazon SES sending limits using an AWS SDK (p. 1879)
  - Get the status of an Amazon SES identity using an AWS SDK (p. 1880)
  - List Amazon SES email templates using an AWS SDK (p. 1882)
  - List Amazon SES identities using an AWS SDK (p. 1883)
  - List Amazon SES receipt filters using an AWS SDK (p. 1886)
  - Send email with Amazon SES using an AWS SDK (p. 1886)
  - Send templated email with Amazon SES using an AWS SDK (p. 1891)
  - Update an Amazon SES email template using an AWS SDK (p. 1894)
  - Verify a domain identity with Amazon SES using an AWS SDK (p. 1895)
  - Verify an email identity with Amazon SES using an AWS SDK (p. 1895)
- [Scenarios for Amazon SES using AWS SDKs \(p. 1897\)](#)
  - Copy Amazon SES email and domain identities from one AWS Region to another using an AWS SDK (p. 1897)
  - Generate credentials to connect to an Amazon SES SMTP endpoint (p. 1903)
  - Verify an email identity and send messages with Amazon SES using an AWS SDK (p. 1904)
- [Cross-service examples for Amazon SES using AWS SDKs \(p. 1912\)](#)
  - Build an Amazon Transcribe streaming app (p. 1912)
  - Create a web application to track DynamoDB data (p. 1913)
  - Create an Amazon Redshift item tracker (p. 1914)
  - Create an Aurora Serverless work item tracker (p. 1915)
  - Detect PPE in images with Amazon Rekognition using an AWS SDK (p. 1918)
  - Detect objects in images with Amazon Rekognition using an AWS SDK (p. 1919)
  - Detect people and objects in a video with Amazon Rekognition using an AWS SDK (p. 1921)
  - Use Step Functions to invoke Lambda functions (p. 1922)

## Actions for Amazon SES using AWS SDKs

The following code examples show how to use Amazon Simple Email Service with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create an Amazon SES receipt filter using an AWS SDK \(p. 1868\)](#)
- [Create an Amazon SES receipt rule using an AWS SDK \(p. 1869\)](#)
- [Create an Amazon SES receipt rule set using an AWS SDK \(p. 1871\)](#)
- [Create an Amazon SES email template using an AWS SDK \(p. 1871\)](#)
- [Delete an Amazon SES receipt filter using an AWS SDK \(p. 1873\)](#)
- [Delete an Amazon SES receipt rule using an AWS SDK \(p. 1874\)](#)
- [Delete an Amazon SES rule set using an AWS SDK \(p. 1874\)](#)
- [Delete an Amazon SES email template using an AWS SDK \(p. 1875\)](#)
- [Delete an Amazon SES identity using an AWS SDK \(p. 1877\)](#)
- [Describe an Amazon SES receipt rule set using an AWS SDK \(p. 1878\)](#)
- [Get an existing Amazon SES email template using an AWS SDK \(p. 1879\)](#)
- [Get Amazon SES sending limits using an AWS SDK \(p. 1879\)](#)
- [Get the status of an Amazon SES identity using an AWS SDK \(p. 1880\)](#)
- [List Amazon SES email templates using an AWS SDK \(p. 1882\)](#)
- [List Amazon SES identities using an AWS SDK \(p. 1883\)](#)
- [List Amazon SES receipt filters using an AWS SDK \(p. 1886\)](#)
- [Send email with Amazon SES using an AWS SDK \(p. 1886\)](#)
- [Send templated email with Amazon SES using an AWS SDK \(p. 1891\)](#)
- [Update an Amazon SES email template using an AWS SDK \(p. 1894\)](#)
- [Verify a domain identity with Amazon SES using an AWS SDK \(p. 1895\)](#)
- [Verify an email identity with Amazon SES using an AWS SDK \(p. 1895\)](#)

### Create an Amazon SES receipt filter using an AWS SDK

The following code example shows how to create an Amazon SES receipt filter that blocks incoming mail from an IP address or range of IP addresses.

Python

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesReceiptHandler:  
    """Encapsulates Amazon SES receipt handling functions."""  
    def __init__(self, ses_client, s3_resource):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        :param s3_resource: A Boto3 Amazon S3 resource.  
        """
```

```
"""
    self.ses_client = ses_client
    self.s3_resource = s3_resource

def create_receipt_filter(self, filter_name, ip_address_or_range, allow):
    """
    Creates a filter that allows or blocks incoming mail from an IP address or
    range.

    :param filter_name: The name to give the filter.
    :param ip_address_or_range: The IP address or range to block or allow.
    :param allow: When True, incoming mail is allowed from the specified IP
                  address or range; otherwise, it is blocked.
    """
    try:
        policy = 'Allow' if allow else 'Block'
        self.ses_client.create_receipt_filter(
            Filter={
                'Name': filter_name,
                'IpFilter': {
                    'Cidr': ip_address_or_range,
                    'Policy': policy}})
        logger.info(
            "Created receipt filter %s to %s IP of %s.", filter_name, policy,
            ip_address_or_range)
    except ClientError:
        logger.exception("Couldn't create receipt filter %s.", filter_name)
        raise
```

- For API details, see [CreateReceiptFilter](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an Amazon SES receipt rule using an AWS SDK

The following code example shows how to create an Amazon SES receipt rule.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an Amazon S3 bucket where Amazon SES can put copies of incoming emails and create a rule that copies incoming email to the bucket for a specific list of recipients.

```
class SesReceiptHandler:
    """
    Encapsulates Amazon SES receipt handling functions.
    """
    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource

    def create_bucket_for_copy(self, bucket_name):
        """
        Creates a bucket that can receive copies of emails from Amazon SES. This
        includes adding a policy to the bucket that grants Amazon SES permission
        to put objects in the bucket.
        """
```

```

:param bucket_name: The name of the bucket to create.
:return: The newly created bucket.
"""
allow_ses_put_policy = {
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "AllowSESPut",
        "Effect": "Allow",
        "Principal": {
            "Service": "ses.amazonaws.com"},
        "Action": "s3:PutObject",
        "Resource": f"arn:aws:s3::{bucket_name}/*"}]}
bucket = None
try:
    bucket = self.s3_resource.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            'LocationConstraint':
                self.s3_resource.meta.client.meta.region_name})
    bucket.wait_until_exists()
    bucket.Policy().put(Policy=json.dumps(allow_ses_put_policy))
    logger.info("Created bucket %s to receive copies of emails.", bucket_name)
except ClientError:
    logger.exception("Couldn't create bucket to receive copies of emails.")
    if bucket is not None:
        bucket.delete()
    raise
else:
    return bucket

def create_s3_copy_rule(
        self, rule_set_name, rule_name, recipients, bucket_name, prefix):
    """
    Creates a rule so that all emails received by the specified recipients are
    copied to an Amazon S3 bucket.

    :param rule_set_name: The name of a previously created rule set to contain
        this rule.
    :param rule_name: The name to give the rule.
    :param recipients: When an email is received by one of these recipients, it
        is copied to the Amazon S3 bucket.
    :param bucket_name: The name of the bucket to receive email copies. This
        bucket must allow Amazon SES to put objects into it.
    :param prefix: An object key prefix to give the emails copied to the
        bucket.
    """
    try:
        self.ses_client.create_receipt_rule(
            RuleSetName=rule_set_name,
            Rule={
                'Name': rule_name,
                'Enabled': True,
                'Recipients': recipients,
                'Actions': [
                    {
                        'S3Action': {
                            'BucketName': bucket_name,
                            'ObjectKeyPrefix': prefix
                        }}]})
        logger.info(
            "Created rule %s to copy mail received by %s to bucket %s.",
            rule_name, recipients, bucket_name)
    except ClientError:
        logger.exception("Couldn't create rule %s.", rule_name)
        raise

```

- For API details, see [CreateReceiptRule](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an Amazon SES receipt rule set using an AWS SDK

The following code example shows how to create an Amazon SES receipt rule set to organize rules applied to incoming emails.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesReceiptHandler:  
    """Encapsulates Amazon SES receipt handling functions."""  
    def __init__(self, ses_client, s3_resource):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        :param s3_resource: A Boto3 Amazon S3 resource.  
        """  
        self.ses_client = ses_client  
        self.s3_resource = s3_resource  
  
    def create_receipt_rule_set(self, rule_set_name):  
        """  
        Creates an empty rule set. Rule sets contain individual rules and can be  
        used to organize rules.  
  
        :param rule_set_name: The name to give the rule set.  
        """  
        try:  
            self.ses_client.create_receipt_rule_set(RuleSetName=rule_set_name)  
            logger.info("Created receipt rule set %s.", rule_set_name)  
        except ClientError:  
            logger.exception("Couldn't create receipt rule set %s.", rule_set_name)  
            raise
```

- For API details, see [CreateReceiptRuleSet](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an Amazon SES email template using an AWS SDK

The following code examples show how to create an Amazon SES email template.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create an email template.
/// </summary>
/// <param name="name">Name of the template.</param>
/// <param name="subject">Email subject.</param>
/// <param name="text">Email body text.</param>
/// <param name="html">Email HTML body text.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CreateEmailTemplateAsync(string name, string subject,
string text,
        string html)
{
    var success = false;
    try
    {
        var response = await _amazonSimpleEmailService.CreateTemplateAsync(
            new CreateTemplateRequest
            {
                Template = new Template
                {
                    TemplateName = name,
                    SubjectPart = subject,
                    TextPart = text,
                    HtmlPart = html
                }
            });
        success = response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (Exception ex)
    {
        Console.WriteLine("CreateEmailTemplateAsync failed with exception: " +
            ex.Message);
    }

    return success;
}
```

- For API details, see [CreateTemplate](#) in *AWS SDK for .NET API Reference*.

## Python

### [SDK for Python \(Boto3\)](#)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

```

```
:param subject: The subject of the email.
:param text: The text version of the email.
:param html: The html version of the email.
"""
self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
logger.info("Extracted template tags: %s", self.template_tags)

def create_template(self, name, subject, text, html):
    """
    Creates an email template.

    :param name: The name of the template.
    :param subject: The subject of the email.
    :param text: The plain text version of the email.
    :param html: The HTML version of the email.
    """
    try:
        template = {
            'TemplateName': name,
            'SubjectPart': subject,
            'TextPart': text,
            'HtmlPart': html}
        self.ses_client.create_template(Template=template)
        logger.info("Created template %s.", name)
        self.template = template
        self._extract_tags(subject, text, html)
    except ClientError:
        logger.exception("Couldn't create template %s.", name)
        raise
```

- For API details, see [CreateTemplate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an Amazon SES receipt filter using an AWS SDK

The following code example shows how to delete an Amazon SES receipt filter.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""
    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource

    def delete_receipt_filter(self, filter_name):
        """
        Deletes a receipt filter.

        :param filter_name: The name of the filter to delete.
```

```
"""
try:
    self.ses_client.delete_receipt_filter(FilterName=filter_name)
    logger.info("Deleted receipt filter %s.", filter_name)
except ClientError:
    logger.exception("Couldn't delete receipt filter %s.", filter_name)
    raise
```

- For API details, see [DeleteReceiptFilter](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an Amazon SES receipt rule using an AWS SDK

The following code example shows how to delete an Amazon SES receipt rule.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""
    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource

    def delete_receipt_rule(self, rule_set_name, rule_name):
        """
        Deletes a rule.

        :param rule_set_name: The rule set that contains the rule to delete.
        :param rule_name: The rule to delete.
        """
        try:
            self.ses_client.delete_receipt_rule(
                RuleSetName=rule_set_name, RuleName=rule_name)
            logger.info("Removed rule %s from rule set %s.", rule_name,
                       rule_set_name)
        except ClientError:
            logger.exception(
                "Couldn't remove rule %s from rule set %s.", rule_name,
                rule_set_name)
            raise
```

- For API details, see [DeleteReceiptRule](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an Amazon SES rule set using an AWS SDK

The following code example shows how to delete an Amazon SES rule set and all of the rules it contains.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesReceiptHandler:  
    """Encapsulates Amazon SES receipt handling functions."""  
    def __init__(self, ses_client, s3_resource):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        :param s3_resource: A Boto3 Amazon S3 resource.  
        """  
        self.ses_client = ses_client  
        self.s3_resource = s3_resource  
  
    def delete_receipt_rule_set(self, rule_set_name):  
        """  
        Deletes a rule set. When a rule set is deleted, all of the rules it  
        contains  
        are also deleted.  
  
        :param rule_set_name: The name of the rule set to delete.  
        """  
        try:  
            self.ses_client.delete_receipt_rule_set(RuleSetName=rule_set_name)  
            logger.info("Deleted rule set %s.", rule_set_name)  
        except ClientError:  
            logger.exception("Couldn't delete rule set %s.", rule_set_name)  
            raise
```

- For API details, see [DeleteReceiptRuleSet in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete an Amazon SES email template using an AWS SDK

The following code examples show how to delete an Amazon SES email template.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Delete an email template.  
/// </summary>  
/// <param name="templateName">Name of the template.</param>  
/// <returns>True if successful.</returns>  
public async Task<bool> DeleteEmailTemplateAsync(string templateName)  
{  
    var success = false;  
    try  
    {
```

```
        var response = await _amazonSimpleEmailService.DeleteTemplateAsync(
            new DeleteTemplateRequest
            {
                TemplateName = templateName
            });
        success = response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (Exception ex)
    {
        Console.WriteLine("DeleteEmailTemplateAsync failed with exception: " +
ex.Message);
    }

    return success;
}
```

- For API details, see [DeleteTemplate](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)

    def delete_template(self):
        """
        Deletes an email template.
        """
        try:

            self.ses_client.delete_template(TemplateName=self.template['TemplateName'])
            logger.info("Deleted template %s.", self.template['TemplateName'])
            self.template = None
            self.template_tags = None
        except ClientError:
            logger.exception(
                "Couldn't delete template %s.", self.template['TemplateName'])
            raise
```

- For API details, see [DeleteTemplate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an Amazon SES identity using an AWS SDK

The following code examples show how to delete an Amazon SES identity.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Delete an email identity.
/// </summary>
/// <param name="identityEmail">The identity email to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteIdentityAsync(string identityEmail)
{
    var success = false;
    try
    {
        var response = await _amazonSimpleEmailService.DeleteIdentityAsync(
            new DeleteIdentityRequest
            {
                Identity = identityEmail
            });
        success = response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (Exception ex)
    {
        Console.WriteLine("DeleteIdentityAsync failed with exception: " +
            ex.Message);
    }

    return success;
}
```

- For API details, see [DeleteIdentity](#) in *AWS SDK for .NET API Reference*.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""
    def __init__(self, ses_client):
        """
```

```
:param ses_client: A Boto3 Amazon SES client.  
"""  
    self.ses_client = ses_client  
  
def delete_identity(self, identity):  
    """  
        Deletes an identity.  
  
    :param identity: The identity to remove.  
    """  
    try:  
        self.ses_client.delete_identity(Identity=identity)  
        logger.info("Deleted identity %s.", identity)  
    except ClientError:  
        logger.exception("Couldn't delete identity %s.", identity)  
        raise
```

- For API details, see [DeleteIdentity](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe an Amazon SES receipt rule set using an AWS SDK

The following code example shows how to describe an Amazon SES receipt rule set.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesReceiptHandler:  
    """Encapsulates Amazon SES receipt handling functions."""  
    def __init__(self, ses_client, s3_resource):  
        """  
            :param ses_client: A Boto3 Amazon SES client.  
            :param s3_resource: A Boto3 Amazon S3 resource.  
        """  
        self.ses_client = ses_client  
        self.s3_resource = s3_resource  
  
    def describe_receipt_rule_set(self, rule_set_name):  
        """  
            Gets data about a rule set.  
  
            :param rule_set_name: The name of the rule set to retrieve.  
            :return: Data about the rule set.  
        """  
        try:  
            response = self.ses_client.describe_receipt_rule_set(  
                RuleSetName=rule_set_name)  
            logger.info("Got data for rule set %s.", rule_set_name)  
        except ClientError:  
            logger.exception("Couldn't get data for rule set %s.", rule_set_name)  
            raise  
        else:  
            return response
```

- For API details, see [DescribeReceiptRuleSet](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get an existing Amazon SES email template using an AWS SDK

The following code example shows how to get an existing Amazon SES email template.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesTemplate:  
    """Encapsulates Amazon SES template functions."""  
    def __init__(self, ses_client):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        """  
        self.ses_client = ses_client  
        self.template = None  
        self.template_tags = set()  
  
    def _extract_tags(self, subject, text, html):  
        """  
        Extracts tags from a template as a set of unique values.  
  
        :param subject: The subject of the email.  
        :param text: The text version of the email.  
        :param html: The html version of the email.  
        """  
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))  
        logger.info("Extracted template tags: %s", self.template_tags)  
  
    def get_template(self, name):  
        """  
        Gets a previously created email template.  
  
        :param name: The name of the template to retrieve.  
        :return: The retrieved email template.  
        """  
        try:  
            response = self.ses_client.get_template(TemplateName=name)  
            self.template = response['Template']  
            logger.info("Got template %s.", name)  
            self._extract_tags(  
                self.template['SubjectPart'], self.template['TextPart'],  
                self.template['HtmlPart'])  
        except ClientError:  
            logger.exception("Couldn't get template %s.", name)  
            raise  
        else:  
            return self.template
```

- For API details, see [GetTemplate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get Amazon SES sending limits using an AWS SDK

The following code example shows how to get Amazon SES sending limits.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get information on the current account's send quota.
/// </summary>
/// <returns>The send quota response data.</returns>
public async Task<GetSendQuotaResponse> GetSendQuotaAsync()
{
    var result = new GetSendQuotaResponse();
    try
    {
        var response = await _amazonSimpleEmailService.GetSendQuotaAsync(
            new GetSendQuotaRequest());
        result = response;
    }
    catch (Exception ex)
    {
        Console.WriteLine("GetSendQuotaAsync failed with exception: " +
            ex.Message);
    }

    return result;
}
```

- For API details, see [GetSendQuota](#) in *AWS SDK for .NET API Reference*.

## Get the status of an Amazon SES identity using an AWS SDK

The following code examples show how to get the status of an Amazon SES identity.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get identity verification status for an email.
/// </summary>
/// <returns>The verification status of the email.</returns>
public async Task<VerificationStatus> GetIdentityStatusAsync(string email)
{
    var result = VerificationStatus.TemporaryFailure;
    try
    {
        var response =
            await
                _amazonSimpleEmailService.GetIdentityVerificationAttributesAsync(
```

```
        new GetIdentityVerificationAttributesRequest
    {
        Identities = new List<string> { email }
    });

    if (response.VerificationAttributes.ContainsKey(email))
        result = response.VerificationAttributes[email].VerificationStatus;
}
catch (Exception ex)
{
    Console.WriteLine("GetIdentityStatusAsync failed with exception: " +
ex.Message);
}

return result;
}
```

- For API details, see [GetIdentityVerificationAttributes](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client

    def get_identity_status(self, identity):
        """
        Gets the status of an identity. This can be used to discover whether
        an identity has been successfully verified.

        :param identity: The identity to query.
        :return: The status of the identity.
        """
        try:
            response = self.ses_client.get_identity_verification_attributes(
                Identities=[identity])
            status = response['VerificationAttributes'].get(
                identity, {'VerificationStatus': 'NotFound'})['VerificationStatus']
            logger.info("Got status of %s for %s.", status, identity)
        except ClientError:
            logger.exception("Couldn't get status for %s.", identity)
            raise
        else:
            return status
```

- For API details, see [GetIdentityVerificationAttributes](#) in *AWS SDK for Python (Boto3) API Reference*.

## List Amazon SES email templates using an AWS SDK

The following code examples show how to list Amazon SES email templates.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// List email templates for the current account.
/// </summary>
/// <returns>A list of template metadata.</returns>
public async Task<List<TemplateMetadata>> ListEmailTemplatesAsync()
{
    var result = new List<TemplateMetadata>();
    try
    {
        var response = await _amazonSimpleEmailService.ListTemplatesAsync(
            new ListTemplatesRequest());
        result = response.TemplatesMetadata;
    }
    catch (Exception ex)
    {
        Console.WriteLine("ListEmailTemplatesAsync failed with exception: " +
            ex.Message);
    }

    return result;
}
```

- For API details, see [ListTemplates](#) in *AWS SDK for .NET API Reference*.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllTemplates(SesV2Client sesv2Client) {

    try {
        ListEmailTemplatesRequest templatesRequest =
ListEmailTemplatesRequest.builder()
            .pageSize(1)
            .build();

        ListEmailTemplatesResponse response =
sesv2Client.listEmailTemplates(templatesRequest);
        response.templatesMetadata().forEach(template ->
            System.out.println("Template name: " +
                template.templateName()));
    }
}
```

```
        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- For API details, see [ListTemplates](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)

    def list_templates(self):
        """
        Gets a list of all email templates for the current account.

        :return: The list of retrieved email templates.
        """
        try:
            response = self.ses_client.list_templates()
            templates = response['TemplatesMetadata']
            logger.info("Got %s templates.", len(templates))
        except ClientError:
            logger.exception("Couldn't get templates.")
            raise
        else:
            return templates
```

- For API details, see [ListTemplates](#) in *AWS SDK for Python (Boto3) API Reference*.

## List Amazon SES identities using an AWS SDK

The following code examples show how to list Amazon SES identities.

.NET

**AWS SDK for .NET**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get the identities of a specified type for the current account.
/// </summary>
/// <param name="identityType">IdentityType to list.</param>
/// <returns>The list of identities.</returns>
public async Task<List<string>> ListIdentitiesAsync(IdentityType identityType)
{
    var result = new List<string>();
    try
    {
        var response = await _amazonSimpleEmailService.ListIdentitiesAsync(
            new ListIdentitiesRequest
            {
                IdentityType = identityType
            });
        result = response.Identities;
    }
    catch (Exception ex)
    {
        Console.WriteLine("ListIdentitiesAsync failed with exception: " +
            ex.Message);
    }

    return result;
}
```

- For API details, see [ListIdentities](#) in [AWS SDK for .NET API Reference](#).

Java

**SDK for Java 2.x**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listSESEmailIdentities(SesClient client) {

    try {
        ListIdentitiesResponse identitiesResponse = client.listIdentities();
        List<String> identities = identitiesResponse.getIdentities();
        for (String identity : identities) {
            System.out.println("The identity is "+identity);
        }
    } catch (SesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- For API details, see [ListIdentities](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import { ListIdentitiesCommand } from "@aws-sdk/client-ses";
import { sesClient } from "./libs/sesClient.js";

const createListIdentitiesCommand = () =>
  new ListIdentitiesCommand({ IdentityType: "EmailAddress", MaxItems: 10 });

const run = async () => {
  const listIdentitiesCommand = createListIdentitiesCommand();

  try {
    return await sesClient.send(listIdentitiesCommand);
  } catch (err) {
    console.log("Failed to list identities.", err);
    return err;
  }
};
```

- For API details, see [ListIdentities](#) in *AWS SDK for JavaScript API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client

    def list_identities(self, identity_type, max_items):
        """
        Gets the identities of the specified type for the current account.

        :param identity_type: The type of identity to retrieve, such as
        EmailAddress.
        :param max_items: The maximum number of identities to retrieve.
        :return: The list of retrieved identities.
        """
        try:
```

```
        response = self.ses_client.list_identities(
            IdentityType=identity_type, MaxItems=max_items)
        identities = response['Identities']
        logger.info("Got %s identities for the current account.", len(identities))
    except ClientError:
        logger.exception("Couldn't list identities for the current account.")
        raise
    else:
        return identities
```

- For API details, see [ListIdentities](#) in *AWS SDK for Python (Boto3) API Reference*.

## List Amazon SES receipt filters using an AWS SDK

The following code example shows how to list Amazon SES receipt filters.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""
    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource

    def list_receipt_filters(self):
        """
        Gets the list of receipt filters for the current account.

        :return: The list of receipt filters.
        """
        try:
            response = self.ses_client.list_receipt_filters()
            filters = response['Filters']
            logger.info("Got %s receipt filters.", len(filters))
        except ClientError:
            logger.exception("Couldn't get receipt filters.")
            raise
        else:
            return filters
```

- For API details, see [ListReceiptFilters](#) in *AWS SDK for Python (Boto3) API Reference*.

## Send email with Amazon SES using an AWS SDK

The following code examples show how to send email with Amazon SES.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Send an email by using Amazon SES.
///</summary>
///<param name="toAddresses">List of recipients.</param>
///<param name="ccAddresses">List of cc recipients.</param>
///<param name="bccAddresses">List of bcc recipients.</param>
///<param name="bodyHtml">Body of the email in HTML.</param>
///<param name="bodyText">Body of the email in plain text.</param>
///<param name="subject">Subject line of the email.</param>
///<param name="senderAddress">From address.</param>
///<returns>The messageId of the email.</returns>
public async Task<string> SendEmailAsync(List<string> toAddresses,
    List<string> ccAddresses, List<string> bccAddresses,
    string bodyHtml, string bodyText, string subject, string senderAddress)
{
    var messageId = "";
    try
    {
        var response = await _amazonSimpleEmailService.SendEmailAsync(
            new SendEmailRequest
            {
                Destination = new Destination
                {
                    BccAddresses = bccAddresses,
                    CcAddresses = ccAddresses,
                    ToAddresses = toAddresses
                },
                Message = new Message
                {
                    Body = new Body
                    {
                        Html = new Content
                        {
                            Charset = "UTF-8",
                            Data = bodyHtml
                        },
                        Text = new Content
                        {
                            Charset = "UTF-8",
                            Data = bodyText
                        }
                    },
                    Subject = new Content
                    {
                        Charset = "UTF-8",
                        Data = subject
                    }
                },
                Source = senderAddress
            });
        messageId = response.MessageId;
    }
    catch (Exception ex)
    {
```

```
        Console.WriteLine("SendEmailAsync failed with exception: " +
    ex.Message);
}

return messageId;
}
```

- For API details, see [SendEmail](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void send(SesClient client,
                      String sender,
                      String recipient,
                      String subject,
                      String bodyHTML
) throws MessagingException {

    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    Content content = Content.builder()
        .data(bodyHTML)
        .build();

    Content sub = Content.builder()
        .data(subject)
        .build();

    Body body = Body.builder()
        .html(content)
        .build();

    Message msg = Message.builder()
        .subject(sub)
        .body(body)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .message(msg)
        .source(sender)
        .build();

    try {
        System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");
        client.sendEmail(emailRequest);

    } catch (SesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}

public static void sendemailAttachment(SesClient client,
                                      String sender,
                                      String recipient,
                                      String subject,
                                      String bodyText,
                                      String bodyHTML,
                                      String fileLocation) throws AddressException,
                                      MessagingException, IOException {
    java.io.File theFile = new java.io.File(fileLocation);
    byte[] fileContent = Files.readAllBytes(theFile.toPath());

    Session session = Session.getDefaultInstance(new Properties());

    // Create a new MimeMessage object.
    MimeMessage message = new MimeMessage(session);

    // Add subject, from and to lines.
    message.setSubject(subject, "UTF-8");
    message.setFrom(new InternetAddress(sender));
    message.setRecipients(Message.RecipientType.TO,
    InternetAddress.parse(recipient));

    // Create a multipart/alternative child container.
    MimeMultipart msgBody = new MimeMultipart("alternative");

    // Create a wrapper for the HTML and text parts.
    MimeBodyPart wrap = new MimeBodyPart();

    // Define the text part.
    MimeBodyPart textPart = new MimeBodyPart();
    textPart.setContent(bodyText, "text/plain; charset=UTF-8");

    // Define the HTML part.
    MimeBodyPart htmlPart = new MimeBodyPart();
    htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");

    // Add the text and HTML parts to the child container.
    msgBody.addBodyPart(textPart);
    msgBody.addBodyPart(htmlPart);

    // Add the child container to the wrapper object.
    wrap.setContent(msgBody);

    // Create a multipart/mixed parent container.
    MimeMultipart msg = new MimeMultipart("mixed");

    // Add the parent container to the message.
    message.setContent(msg);
    msg.addBodyPart(wrap);

    // Define the attachment.
    MimeBodyPart att = new MimeBodyPart();
    DataSource fds = new ByteArrayDataSource(fileContent, "application/
vnd.openxmlformats-officedocument.spreadsheetml.sheet");
    att.setDataHandler(new DataHandler(fds));

    String reportName = "WorkReport.xls";
    att.setFileName(reportName);

    // Add the attachment to the message.
    msg.addBodyPart(att);

    try {
```

```
System.out.println("Attempting to send an email through Amazon SES " +  
"using the AWS SDK for Java...");  
  
ByteArrayOutputStream outputStream = new ByteArrayOutputStream();  
message.writeTo(outputStream);  
  
ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());  
  
byte[] arr = new byte[buf.remaining()];  
buf.get(arr);  
  
SdkBytes data = SdkBytes.fromByteArray(arr);  
RawMessage rawMessage = RawMessage.builder()  
    .data(data)  
    .build();  
  
SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()  
    .rawMessage(rawMessage)  
    .build();  
  
client.sendRawEmail(rawEmailRequest);  
  
} catch (SesException e) {  
    System.err.println(e.awsErrorDetails().errorMessage());  
    System.exit(1);  
}  
System.out.println("Email sent using SesClient with attachment");  
}
```

- For API details, see [SendEmail](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesMailSender:  
    """Encapsulates functions to send emails with Amazon SES."""  
    def __init__(self, ses_client):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        """  
        self.ses_client = ses_client  
  
    def send_email(self, source, destination, subject, text, html, reply_tos=None):  
        """  
        Sends an email.  
  
        Note: If your account is in the Amazon SES sandbox, the source and  
        destination email accounts must both be verified.  
  
        :param source: The source email account.  
        :param destination: The destination email account.  
        :param subject: The subject of the email.  
        :param text: The plain text version of the body of the email.  
        :param html: The HTML version of the body of the email.  
        :param reply_tos: Email accounts that will receive a reply if the recipient  
            replies to the message.  
        :return: The ID of the message, assigned by Amazon SES.  
        """
```

```
"""
send_args = {
    'Source': source,
    'Destination': destination.to_service_format(),
    'Message': {
        'Subject': {'Data': subject},
        'Body': {'Text': {'Data': text}, 'Html': {'Data': html}}}}
if reply_tos is not None:
    send_args['ReplyToAddresses'] = reply_tos
try:
    response = self.ses_client.send_email(**send_args)
    message_id = response['MessageId']
    logger.info(
        "Sent mail %s from %s to %s.", message_id, source, destination.tos)
except ClientError:
    logger.exception(
        "Couldn't send mail from %s to %s.", source, destination.tos)
    raise
else:
    return message_id
```

- For API details, see [SendEmail](#) in *AWS SDK for Python (Boto3) API Reference*.

## Send templated email with Amazon SES using an AWS SDK

The following code examples show how to send templated email with Amazon SES.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Send an email using a template.
/// </summary>
/// <param name="sender">Address of the sender.</param>
/// <param name="recipients">Addresses of the recipients.</param>
/// <param name="templateName">Name of the email template.</param>
/// <param name="templateDataObject">Data for the email template.</param>
/// <returns>The messageId of the email.</returns>
public async Task<string> SendTemplateEmailAsync(string sender, List<string>
recipients,
    string templateName, object templateDataObject)
{
    var messageId = "";
    try
    {
        // Template data should be serialized JSON from either a class or a
dynamic object.
        var templateData = JsonSerializer.Serialize(templateDataObject);

        var response = await _amazonSimpleEmailService.SendTemplatedEmailAsync(
            new SendTemplatedEmailRequest
            {
                Source = sender,
                Destination = new Destination
```

```
        {
            ToAddresses = recipients
        },
        Template = templateName,
        TemplateData = templateData
    });
    messageId = response.MessageId;
}
catch (Exception ex)
{
    Console.WriteLine("SendTemplateEmailAsync failed with exception: " +
ex.Message);
}

return messageId;
}
```

- For API details, see [SendTemplatedEmail](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void send(SesV2Client client, String sender, String recipient,
String templateName){

    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    /*
        Specify both name and favorite animal (favoriteanimal) in your code when
        defining the Template object.
        If you don't specify all the variables in the template, Amazon SES doesn't
        send the email.
    */
    Template myTemplate = Template.builder()
        .templateName(templateName)
        .templateData("{\n" +
            " \"name\": \"Jason\"\n" +
            " \"favoriteanimal\": \"Cat\"\n" +
            "}")
        .build();

    EmailContent emailContent = EmailContent.builder()
        .template(myTemplate)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .content(emailContent)
        .fromEmailAddress(sender)
        .build();

    try {
        System.out.println("Attempting to send an email based on a template
using the AWS SDK for Java (v2)...");
    }
}
```

```
        client.sendEmail(emailRequest);
        System.out.println("email based on a template was sent");

    } catch (SesV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SendTemplatedEmail in AWS SDK for Java 2.x API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client

    def send_templated_email(
            self, source, destination, template_name, template_data,
            reply_to=None):
        """
        Sends an email based on a template. A template contains replaceable tags
        each enclosed in two curly braces, such as {{name}}. The template data
        passed
        in this function contains key-value pairs that define the values to insert
        in place of the template tags.

        Note: If your account is in the Amazon SES sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param template_name: The name of a previously created template.
        :param template_data: JSON-formatted key-value pairs of replacement values
            that are inserted in the template before it is sent.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            'Source': source,
            'Destination': destination.to_service_format(),
            'Template': template_name,
            'TemplateData': json.dumps(template_data)
        }
        if reply_to is not None:
            send_args['ReplyToAddresses'] = reply_to
        try:
            response = self.ses_client.send_templated_email(**send_args)
            message_id = response['MessageId']
            logger.info(
                "Sent templated mail %s from %s to %s.", message_id, source,
                destination.tos)
        except ClientError:
```

```
        logger.exception(
            "Couldn't send templated mail from %s to %s.", source,
destination.tos)
        raise
    else:
        return message_id
```

- For API details, see [SendTemplatedEmail](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update an Amazon SES email template using an AWS SDK

The following code example shows how to update an Amazon SES email template.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)

    def update_template(self, name, subject, text, html):
        """
        Updates a previously created email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                'TemplateName': name,
                'SubjectPart': subject,
                'TextPart': text,
                'HtmlPart': html}
            self.ses_client.update_template(Template=template)
            logger.info("Updated template %s.", name)
            self.template = template
        except ClientError as e:
            logger.error(e.response['Error']['Message'])
```

```
        self._extract_tags(subject, text, html)
    except ClientError:
        logger.exception("Couldn't update template %s.", name)
        raise
```

- For API details, see [UpdateTemplate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Verify a domain identity with Amazon SES using an AWS SDK

The following code example shows how to verify a domain identity with Amazon SES.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client

    def verify_domain_identity(self, domain_name):
        """
        Starts verification of a domain identity. To complete verification, you
        must
        create a TXT record with a specific format through your DNS provider.

        For more information, see *Verifying a domain with Amazon SES* in the
        Amazon SES documentation:
            https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domain-
        procedure.html

        :param domain_name: The name of the domain to verify.
        :return: The token to include in the TXT record with your DNS provider.
        """
        try:
            response = self.ses_client.verify_domain_identity(Domain=domain_name)
            token = response['VerificationToken']
            logger.info("Got domain verification token for %s.", domain_name)
        except ClientError:
            logger.exception("Couldn't verify domain %s.", domain_name)
            raise
        else:
            return token
```

- For API details, see [VerifyDomainIdentity](#) in *AWS SDK for Python (Boto3) API Reference*.

## Verify an email identity with Amazon SES using an AWS SDK

The following code examples show how to verify an email identity with Amazon SES.

.NET

**AWS SDK for .NET**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Starts verification of an email identity. This request sends an email
/// from Amazon SES to the specified email address. To complete
/// verification, follow the instructions in the email.
/// </summary>
/// <param name="recipientEmailAddress">Email address to verify.</param>
/// <returns>True if successful.</returns>
public async Task<bool> VerifyEmailIdentityAsync(string recipientEmailAddress)
{
    var success = false;
    try
    {
        var response = await
_amazonSimpleEmailService.VerifyEmailIdentityAsync(
            new VerifyEmailIdentityRequest
            {
                EmailAddress = recipientEmailAddress
            });

        success = response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (Exception ex)
    {
        Console.WriteLine("VerifyEmailIdentityAsync failed with exception: " +
ex.Message);
    }

    return success;
}
```

- For API details, see [VerifyEmailIdentity](#) in *AWS SDK for .NET API Reference*.

Python

**SDK for Python (Boto3)**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client

    def verify_email_identity(self, email_address):
```

```
"""
Starts verification of an email identity. This function causes an email
to be sent to the specified email address from Amazon SES. To complete
verification, follow the instructions in the email.

:param email_address: The email address to verify.
"""
try:
    self.ses_client.verify_email_identity(EmailAddress=email_address)
    logger.info("Started verification of %s.", email_address)
except ClientError:
    logger.exception("Couldn't start verification of %s.", email_address)
    raise
```

- For API details, see [VerifyEmailIdentity](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios for Amazon SES using AWS SDKs

The following code examples show how to use Amazon Simple Email Service with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Copy Amazon SES email and domain identities from one AWS Region to another using an AWS SDK \(p. 1897\)](#)
- [Generate credentials to connect to an Amazon SES SMTP endpoint \(p. 1903\)](#)
- [Verify an email identity and send messages with Amazon SES using an AWS SDK \(p. 1904\)](#)

## Copy Amazon SES email and domain identities from one AWS Region to another using an AWS SDK

The following code example shows how to copy Amazon SES email and domain identities from one AWS Region to another. When domain identities are managed by Route 53, verification records are copied to the domain for the destination Region.

### Python

#### [SDK for Python \(Boto3\)](#)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import argparse
import json
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def get_identities(ses_client):
```

```
"""
Gets the identities for the current Region. The Region is specified in the
Boto3 Amazon SES client object.

:param ses_client: A Boto3 Amazon SES client.
:return: The list of email identities and the list of domain identities.
"""
email_identities = []
domain_identities = []
try:
    identityPaginator = ses_client.getPaginator('list_identities')
    identityIterator = identityPaginator.paginate(
        PaginationConfig={'PageSize': 20})
    for identityPage in identityIterator:
        for identity in identityPage['Identities']:
            if '@' in identity:
                email_identities.append(identity)
            else:
                domain_identities.append(identity)
    logger.info(
        "Found %s email and %s domain identities.", len(email_identities),
        len(domain_identities))
except ClientError:
    logger.exception("Couldn't get identities.")
    raise
else:
    return email_identities, domain_identities

def verify_emails(email_list, ses_client):
    """
    Starts verification of a list of email addresses. Verification causes an email
    to be sent to each address. To complete verification, the recipient must follow
    the instructions in the email.

    :param email_list: The list of email addresses to verify.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of emails that were successfully submitted for verification.
    """
    verified_emails = []
    for email in email_list:
        try:
            ses_client.verify_email_identity(EmailAddress=email)
            verified_emails.append(email)
            logger.info("Started verification of %s.", email)
        except ClientError:
            logger.warning("Couldn't start verification of %s.", email)
    return verified_emails

def verify_domains(domain_list, ses_client):
    """
    Starts verification for a list of domain identities. This returns a token for
    each domain, which must be registered as a TXT record with the DNS provider for
    the domain.

    :param domain_list: The list of domains to verify.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The generated domain tokens to use to completed verification.
    """
    domain_tokens = {}
    for domain in domain_list:
        try:
            response = ses_client.verify_domain_identity(Domain=domain)
            token = response['VerificationToken']
            domain_tokens[domain] = token
        except ClientError:
            logger.warning("Couldn't start verification of %s.", domain)
```

```
        logger.info("Got verification token %s for domain %s.", token, domain)
    except ClientError:
        logger.warning("Couldn't get verification token for domain %s.",
domain)
    return domain_tokens

def get_hosted_zones(route53_client):
    """
    Gets the Amazon Route 53 hosted zones for the current account.

    :param route53_client: A Boto3 Route 53 client.
    :return: The list of hosted zones.
    """
    zones = []
    try:
        zonePaginator = route53_client.get_paginator('list_hosted_zones')
        zoneIterator = zonePaginator.paginate(PaginationConfig={'PageSize': 20})
        zones = [
            zone for zonePage in zoneIterator for zone in
zonePage['HostedZones']]
        logger.info("Found %s hosted zones.", len(zones))
    except ClientError:
        logger.warning("Couldn't get hosted zones.")
    return zones

def find_domain_zone_matches(domains, zones):
    """
    Finds matches between Amazon SES verified domains and Route 53 hosted zones.
    Subdomain matches are taken when found, otherwise root domain matches are
    taken.

    :param domains: The list of domains to match.
    :param zones: The list of hosted zones to match.
    :return: The set of matched domain-zone pairs. When a match is not found, the
            domain is included in the set with a zone value of None.
    """
    domain_zones = {}
    for domain in domains:
        domain_zones[domain] = None
        # Start at the most specific sub-domain and walk up to the root domain
until a
        # zone match is found.
        domain_split = domain.split('.')
        for index in range(0, len(domain_split) - 1):
            sub_domain = '.'.join(domain_split[index:])
            for zone in zones:
                # Normalize the zone name from Route 53 by removing the trailing
'..
                zone_name = zone['Name'][:-1]
                if sub_domain == zone_name:
                    domain_zones[domain] = zone
                    break
            if domain_zones[domain] is not None:
                break
    return domain_zones

def add_route53_verification_record(domain, token, zone, route53_client):
    """
    Adds a domain verification TXT record to the specified Route 53 hosted zone.
    When a TXT record already exists in the hosted zone for the specified domain,
    the existing values are preserved and the new token is added to the list.

    :param domain: The domain to add.
    """
```

```

:param token: The verification token for the domain.
:param zone: The hosted zone where the domain verification record is added.
:param route53_client: A Boto3 Route 53 client.
"""
domain_token_record_set_name = f'_amazones.{domain}'
record_setPaginator = route53_client.getPaginator(
    'list_resource_record_sets')
record_setIterator = record_setPaginator.paginate(
    HostedZoneId=zone['Id'], PaginationConfig={'PageSize': 20})
records = []
for record_set_page in record_setIterator:
    try:
        txt_record_set = next(
            record_set for record_set
            in record_set_page['ResourceRecordSets']
            if record_set['Name'][:-1] == domain_token_record_set_name and
            record_set['Type'] == 'TXT')
        records = txt_record_set['ResourceRecords']
        logger.info(
            "Existing TXT record found in set %s for zone %s.",
            domain_token_record_set_name, zone['Name'])
        break
    except StopIteration:
        pass
records.append({'Value': json.dumps(token)})
changes = [{
    'Action': 'UPSERT',
    'ResourceRecordSet': {
        'Name': domain_token_record_set_name,
        'Type': 'TXT',
        'TTL': 1800,
        'ResourceRecords': records}]}
try:
    route53_client.change_resource_record_sets(
        HostedZoneId=zone['Id'], ChangeBatch={'Changes': changes})
    logger.info(
        "Created or updated the TXT record in set %s for zone %s.",
        domain_token_record_set_name, zone['Name'])
except ClientError as err:
    logger.warning(
        "Got error %s. Couldn't create or update the TXT record for zone %s.",
        err.response['Error']['Code'], zone['Name'])

def generate_dkim_tokens(domain, ses_client):
"""
Generates DKIM tokens for a domain. These must be added as CNAME records to the
DNS provider for the domain.

:param domain: The domain to generate tokens for.
:param ses_client: A Boto3 Amazon SES client.
:return: The list of generated DKIM tokens.
"""
dkim_tokens = []
try:
    dkim_tokens = ses_client.verify_domain_dkim(Domain=domain)['DkimTokens']
    logger.info("Generated %s DKIM tokens for domain %s.", len(dkim_tokens),
domain)
except ClientError:
    logger.warning("Couldn't generate DKIM tokens for domain %s.", domain)
return dkim_tokens

def add_dkim_domain_tokens(hosted_zone, domain, tokens, route53_client):
"""
Adds DKIM domain token CNAME records to a Route 53 hosted zone.

```

```

:param hosted_zone: The hosted zone where the records are added.
:param domain: The domain to add.
:param tokens: The DKIM tokens for the domain to add.
:param route53_client: A Boto3 Route 53 client.
"""
try:
    changes = [
        {
            'Action': 'UPSERT',
            'ResourceRecordSet': {
                'Name': f'{token}._domainkey.{domain}',
                'Type': 'CNAME',
                'TTL': 1800,
                'ResourceRecords': [{'Value': f'{token}.dkim.amazonaws.com'}]
            } for token in tokens]
    route53_client.change_resource_record_sets(
        HostedZoneId=hosted_zone['Id'], ChangeBatch={'Changes': changes})
    logger.info(
        "Added %s DKIM CNAME records to %s in zone %s.", len(tokens),
        domain, hosted_zone['Name'])
except ClientError:
    logger.warning(
        "Couldn't add DKIM CNAME records for %s to zone %s.", domain,
        hosted_zone['Name'])

def configure_sns_topics(identity, topics, ses_client):
    """
    Configures Amazon Simple Notification Service (Amazon SNS) notifications for
    an identity. The Amazon SNS topics must already exist.

    :param identity: The identity to configure.
    :param topics: The list of topics to configure. The choices are Bounce,
    Delivery,
        or Complaint.
    :param ses_client: A Boto3 Amazon SES client.
    """
    for topic in topics:
        topic_arn = input(
            f"Enter the Amazon Resource Name (ARN) of the {topic} topic or press "
            f"Enter to skip: ")
        if topic_arn != '':
            try:
                ses_client.set_identity_notification_topic(
                    Identity=identity, NotificationType=topic, SnsTopic=topic_arn)
                logger.info("Configured %s for %s notifications.", identity, topic)
            except ClientError:
                logger.warning(
                    "Couldn't configure %s for %s notifications.", identity, topic)

def replicate(source_client, destination_client, route53_client):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print(f'Replicating Amazon SES identities and other configuration from '
          f'{source_client.meta.region_name} to '
          f'{destination_client.meta.region_name}.')
    print('*'*88)

    print(f'Retrieving identities from {source_client.meta.region_name}.')
    source_emails, source_domains = get_identities(source_client)
    print('Email addresses found:')
    print(*source_emails)
    print('Domains found:')
    print(*source_domains)

```

```
print("Starting verification for email identities.")
dest_emails = verify_emails(source_emails, destination_client)
print("Getting domain tokens for domain identities.")
dest_domain_tokens = verify_domains(source_domains, destination_client)

# Get Route 53 hosted zones and match them with Amazon SES domains.
answer = input(
    "Is the DNS configuration for your domains managed by Amazon Route 53 (y/n)? ")
use_route53 = answer.lower() == 'y'
hosted_zones = get_hosted_zones(route53_client) if use_route53 else []
if use_route53:
    print("Adding or updating Route 53 TXT records for your domains.")
    domain_zones = find_domain_zone_matches(dest_domain_tokens.keys(),
                                              hosted_zones)
    for domain in domain_zones:
        add_route53_verification_record(
            domain, dest_domain_tokens[domain], domain_zones[domain],
            route53_client)
else:
    print("Use these verification tokens to create TXT records through your DNS provider:")
    pprint(dest_domain_tokens)

answer = input("Do you want to configure DKIM signing for your identities (y/n)? ")
if answer.lower() == 'y':
    # Build a set of unique domains from email and domain identities.
    domains = {email.split('@')[1] for email in dest_emails}
    domains.update(dest_domain_tokens)
    domain_zones = find_domain_zone_matches(domains, hosted_zones)
    for domain, zone in domain_zones.items():
        answer = input(
            f"Do you want to configure DKIM signing for {domain} (y/n)? ")
        if answer.lower() == 'y':
            dkim_tokens = generate_dkim_tokens(domain, destination_client)
            if use_route53 and zone is not None:
                add_dkim_domain_tokens(zone, domain, dkim_tokens,
                                      route53_client)
            else:
                print(
                    "Add the following DKIM tokens as CNAME records through your DNS provider:")
                print(*dkim_tokens, sep='\n')

answer = input(
    "Do you want to configure Amazon SNS notifications for your identities (y/n)? ")
if answer.lower() == 'y':
    for identity in dest_emails + list(dest_domain_tokens.keys()):
        answer = input(
            f"Do you want to configure Amazon SNS topics for {identity} (y/n)? ")
        if answer.lower() == 'y':
            configure sns topics(
                identity, ['Bounce', 'Delivery', 'Complaint'],
                destination_client)

print(f"Replication complete for {destination_client.meta.region_name}.")
print('*'*88)

def main():
```

```
boto3_session = boto3.Session()
ses_regions = boto3_session.get_available_regions('ses')
parser = argparse.ArgumentParser(
    description="Copies email address and domain identities from one AWS Region
to "
                    "another. Optionally adds records for domain verification and
DKIM "
                    "signing to domains that are managed by Amazon Route 53, "
                    "and sets up Amazon SNS notifications for events of interest.")
parser.add_argument(
    'source_region', choices=ses_regions, help="The region to copy from.")
parser.add_argument(
    'destination_region', choices=ses_regions, help="The region to copy to.")
args = parser.parse_args()
source_client = boto3.client('ses', region_name=args.source_region)
destination_client = boto3.client('ses', region_name=args.destination_region)
route53_client = boto3.client('route53')
replicate(source_client, destination_client, route53_client)

if __name__ == '__main__':
    main()
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [ListIdentities](#)
  - [SetIdentityNotificationTopic](#)
  - [VerifyDomainDkim](#)
  - [VerifyDomainIdentity](#)
  - [VerifyEmailIdentity](#)

## Generate credentials to connect to an Amazon SES SMTP endpoint

The following code example shows how to generate credentials to connect to an Amazon SES SMTP endpoint.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/usr/bin/env python3

import hmac
import hashlib
import base64
import argparse

SMTP_REGIONS = [
    'us-east-2',      # US East (Ohio)
    'us-east-1',      # US East (N. Virginia)
    'us-west-2',      # US West (Oregon)
    'ap-south-1',     # Asia Pacific (Mumbai)
    'ap-northeast-2', # Asia Pacific (Seoul)
```

```

'ap-southeast-1',    # Asia Pacific (Singapore)
'ap-southeast-2',    # Asia Pacific (Sydney)
'ap-northeast-1',    # Asia Pacific (Tokyo)
'ca-central-1',      # Canada (Central)
'eu-central-1',      # Europe (Frankfurt)
'eu-west-1',          # Europe (Ireland)
'eu-west-2',          # Europe (London)
'sa-east-1',          # South America (Sao Paulo)
'us-gov-west-1',      # AWS GovCloud (US)
]

# These values are required to calculate the signature. Do not change them.
DATE = "11111111"
SERVICE = "ses"
MESSAGE = "SendRawEmail"
TERMINAL = "aws4_request"
VERSION = 0x04

def sign(key, msg):
    return hmac.new(key, msg.encode('utf-8'), hashlib.sha256).digest()

def calculate_key(secret_access_key, region):
    if region not in SMTP_REGIONS:
        raise ValueError(f"The {region} Region doesn't have an SMTP endpoint.")

    signature = sign(("AWS4" + secret_access_key).encode('utf-8'), DATE)
    signature = sign(signature, region)
    signature = sign(signature, SERVICE)
    signature = sign(signature, TERMINAL)
    signature = sign(signature, MESSAGE)
    signature_and_version = bytes([VERSION]) + signature
    smtp_password = base64.b64encode(signature_and_version)
    return smtp_password.decode('utf-8')

def main():
    parser = argparse.ArgumentParser(
        description='Convert a Secret Access Key for an IAM user to an SMTP password.')
    parser.add_argument(
        'secret', help='The Secret Access Key to convert.')
    parser.add_argument(
        'region',
        help='The AWS Region where the SMTP password will be used.',
        choices=SMTP_REGIONS)
    args = parser.parse_args()
    print(calculate_key(args.secret, args.region))

if __name__ == '__main__':
    main()

```

## Verify an email identity and send messages with Amazon SES using an AWS SDK

The following code example shows how to:

- Add and verify an email address with Amazon SES.
- Send a standard email message.

- Create a template and send a templated email message.
- Send a message by using an Amazon SES SMTP server.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions to wrap Amazon SES identity actions.

```
class SesIdentity:  
    """Encapsulates Amazon SES identity functions."""  
    def __init__(self, ses_client):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        """  
        self.ses_client = ses_client  
  
    def verify_domain_identity(self, domain_name):  
        """  
        Starts verification of a domain identity. To complete verification, you  
        must  
        create a TXT record with a specific format through your DNS provider.  
  
        For more information, see *Verifying a domain with Amazon SES* in the  
        Amazon SES documentation:  
            https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domain-  
            procedure.html  
  
        :param domain_name: The name of the domain to verify.  
        :return: The token to include in the TXT record with your DNS provider.  
        """  
        try:  
            response = self.ses_client.verify_domain_identity(Domain=domain_name)  
            token = response['VerificationToken']  
            logger.info("Got domain verification token for %s.", domain_name)  
        except ClientError:  
            logger.exception("Couldn't verify domain %s.", domain_name)  
            raise  
        else:  
            return token  
  
    def verify_email_identity(self, email_address):  
        """  
        Starts verification of an email identity. This function causes an email  
        to be sent to the specified email address from Amazon SES. To complete  
        verification, follow the instructions in the email.  
  
        :param email_address: The email address to verify.  
        """  
        try:  
            self.ses_client.verify_email_identity(EmailAddress=email_address)  
            logger.info("Started verification of %s.", email_address)  
        except ClientError:  
            logger.exception("Couldn't start verification of %s.", email_address)  
            raise  
  
    def wait_until_identity_exists(self, identity):  
        """  
        Waits until an identity exists. The waiter polls Amazon SES until the
```

```
    identity has been successfully verified or until it exceeds its maximum
time.

    :param identity: The identity to wait for.
    """
    try:
        waiter = self.ses_client.get_waiter('identity_exists')
        logger.info("Waiting until %s exists.", identity)
        waiter.wait(Identities=[identity])
    except WaiterError:
        logger.error("Waiting for identity %s failed or timed out.", identity)
        raise

def get_identity_status(self, identity):
    """
    Gets the status of an identity. This can be used to discover whether
    an identity has been successfully verified.

    :param identity: The identity to query.
    :return: The status of the identity.
    """
    try:
        response = self.ses_client.get_identity_verification_attributes(
            Identities=[identity])
        status = response['VerificationAttributes'].get(
            identity, {'VerificationStatus': 'NotFound'})['VerificationStatus']
        logger.info("Got status of %s for %s.", status, identity)
    except ClientError:
        logger.exception("Couldn't get status for %s.", identity)
        raise
    else:
        return status

def delete_identity(self, identity):
    """
    Deletes an identity.

    :param identity: The identity to remove.
    """
    try:
        self.ses_client.delete_identity(Identity=identity)
        logger.info("Deleted identity %s.", identity)
    except ClientError:
        logger.exception("Couldn't delete identity %s.", identity)
        raise

def list_identities(self, identity_type, max_items):
    """
    Gets the identities of the specified type for the current account.

    :param identity_type: The type of identity to retrieve, such as
    EmailAddress.
    :param max_items: The maximum number of identities to retrieve.
    :return: The list of retrieved identities.
    """
    try:
        response = self.ses_client.list_identities(
            IdentityType=identity_type, MaxItems=max_items)
        identities = response['Identities']
        logger.info("Got %s identities for the current account.", len(identities))
    except ClientError:
        logger.exception("Couldn't list identities for the current account.")
        raise
    else:
        return identities
```

Create functions to wrap Amazon SES template actions.

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)

    def create_template(self, name, subject, text, html):
        """
        Creates an email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                'TemplateName': name,
                'SubjectPart': subject,
                'TextPart': text,
                'HtmlPart': html}
            self.ses_client.create_template(Template=template)
            logger.info("Created template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't create template %s.", name)
            raise

    def delete_template(self):
        """
        Deletes an email template.
        """
        try:
            self.ses_client.delete_template(TemplateName=self.template['TemplateName'])
            logger.info("Deleted template %s.", self.template['TemplateName'])
            self.template = None
            self.template_tags = None
        except ClientError:
            logger.exception(
                "Couldn't delete template %s.", self.template['TemplateName'])
            raise

    def get_template(self, name):
        """
```

```

Gets a previously created email template.

:param name: The name of the template to retrieve.
:return: The retrieved email template.
"""
try:
    response = self.ses_client.get_template(TemplateName=name)
    self.template = response['Template']
    logger.info("Got template %s.", name)
    self._extract_tags(
        self.template['SubjectPart'], self.template['TextPart'],
        self.template['HtmlPart'])
except ClientError:
    logger.exception("Couldn't get template %s.", name)
    raise
else:
    return self.template

def list_templates(self):
"""
Gets a list of all email templates for the current account.

:return: The list of retrieved email templates.
"""
try:
    response = self.ses_client.list_templates()
    templates = response['TemplatesMetadata']
    logger.info("Got %s templates.", len(templates))
except ClientError:
    logger.exception("Couldn't get templates.")
    raise
else:
    return templates

def update_template(self, name, subject, text, html):
"""
Updates a previously created email template.

:param name: The name of the template.
:param subject: The subject of the email.
:param text: The plain text version of the email.
:param html: The HTML version of the email.
"""
try:
    template = {
        'TemplateName': name,
        'SubjectPart': subject,
        'TextPart': text,
        'HtmlPart': html}
    self.ses_client.update_template(Template=template)
    logger.info("Updated template %s.", name)
    self.template = template
    self._extract_tags(subject, text, html)
except ClientError:
    logger.exception("Couldn't update template %s.", name)
    raise

```

Create functions to wrap Amazon SES email actions.

```

class SesDestination:
    """Contains data about an email destination."""
    def __init__(self, tos, ccs=None, bccs=None):
        """
        :param tos: The list of recipients on the 'To:' line.

```

```
:param ccs: The list of recipients on the 'CC:' line.
:param bccs: The list of recipients on the 'BCC:' line.
"""
self.tos = tos
self.ccs = ccs
self.bccs = bccs

def to_service_format(self):
"""
:return: The destination data in the format expected by Amazon SES.
"""
svc_format = {'ToAddresses': self.tos}
if self.ccs is not None:
    svc_format['CcAddresses'] = self.ccs
if self.bccs is not None:
    svc_format['BccAddresses'] = self.bccs
return svc_format

class SesMailSender:
"""Encapsulates functions to send emails with Amazon SES."""
def __init__(self, ses_client):
"""
:param ses_client: A Boto3 Amazon SES client.
"""
self.ses_client = ses_client

def send_email(self, source, destination, subject, text, html, reply_tos=None):
"""
Sends an email.

Note: If your account is in the Amazon SES sandbox, the source and
destination email accounts must both be verified.

:param source: The source email account.
:param destination: The destination email account.
:param subject: The subject of the email.
:param text: The plain text version of the body of the email.
:param html: The HTML version of the body of the email.
:param reply_tos: Email accounts that will receive a reply if the recipient
replies to the message.
:return: The ID of the message, assigned by Amazon SES.
"""
send_args = {
    'Source': source,
    'Destination': destination.to_service_format(),
    'Message': {
        'Subject': {'Data': subject},
        'Body': {'Text': {'Data': text}, 'Html': {'Data': html}}}}
if reply_tos is not None:
    send_args['ReplyToAddresses'] = reply_tos
try:
    response = self.ses_client.send_email(**send_args)
    message_id = response['MessageId']
    logger.info(
        "Sent mail %s from %s to %s.", message_id, source, destination.tos)
except ClientError:
    logger.exception(
        "Couldn't send mail from %s to %s.", source, destination.tos)
    raise
else:
    return message_id

def send_templated_email(
    self, source, destination, template_name, template_data,
    reply_tos=None):
"""


```

Sends an email based on a template. A template contains replaceable tags each enclosed in two curly braces, such as {{name}}. The template data passed in this function contains key-value pairs that define the values to insert in place of the template tags.

Note: If your account is in the Amazon SES sandbox, the source and destination email accounts must both be verified.

```

:param source: The source email account.
:param destination: The destination email account.
:param template_name: The name of a previously created template.
:param template_data: JSON-formatted key-value pairs of replacement values
                      that are inserted in the template before it is sent.
:return: The ID of the message, assigned by Amazon SES.
"""
send_args = {
    'Source': source,
    'Destination': destination.to_service_format(),
    'Template': template_name,
    'TemplateData': json.dumps(template_data)
}
if reply_tos is not None:
    send_args['ReplyToAddresses'] = reply_tos
try:
    response = self.ses_client.send templated_email(**send_args)
    message_id = response['MessageId']
    logger.info(
        "Sent templated mail %s from %s to %s.", message_id, source,
        destination.tos)
except ClientError:
    logger.exception(
        "Couldn't send templated mail from %s to %s.", source,
        destination.tos)
    raise
else:
    return message_id

```

Verify an email address with Amazon SES and send messages.

```

def usage_demo():
    print('*'*88)
    print("Welcome to the Amazon Simple Email Service (Amazon SES) email demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    ses_client = boto3.client('ses')
    ses_identity = SesIdentity(ses_client)
    ses_mail_sender = SesMailSender(ses_client)
    ses_template = SesTemplate(ses_client)
    email = input(
        "Enter an email address to send mail with Amazon SES: ")
    status = ses_identity.get_identity_status(email)
    verified = status == 'Success'
    if not verified:
        answer = input(
            f"The address '{email}' is not verified with Amazon SES. Unless your "
            f"Amazon SES account is out of sandbox, you can send mail only from "
            f"and to verified accounts. Do you want to verify this account for use "
        )
        if answer.lower() == 'y':

```

```

        ses_identity.verify_email_identity(email)
        print(f"Follow the steps in the email to {email} to complete
verification.")
        print("Waiting for verification...")
        try:
            ses_identity.wait_until_identity_exists(email)
            print(f"Identity verified for {email}.")
            verified = True
        except WaiterError:
            print(f"Verification timeout exceeded. You must complete the "
                  f"steps in the email sent to {email} to verify the address.")

    if verified:
        test_message_text = "Hello from the Amazon SES mail demo!"
        test_message_html = "<p>Hello!</p><p>From the <b>Amazon SES</b> mail demo!
</p>"
        print(f"Sending mail from {email} to {email}.")
        ses_mail_sender.send_email(
            email, SesDestination([email]), "Amazon SES demo",
            test_message_text, test_message_html)
        input("Mail sent. Check your inbox and press Enter to continue.")

        template = {
            'name': 'doc-example-template',
            'subject': 'Example of an email template.',
            'text': "This is what {{name}} will {{action}} if {{name}} can't
display "
                    "HTML.",
            'html': "<p><i>This</i> is what {{name}} will {{action}} if {{name}} "
                    "<b>can</b> display HTML.</p>"}
        print("Creating a template and sending a templated email.")
        ses_template.create_template(**template)
        template_data = {'name': email.split('@')[0], 'action': 'read'}
        if ses_template.verify_tags(template_data):
            ses_mail_sender.send_templated_email(
                email, SesDestination([email]), ses_template.name(), template_data)
            input("Mail sent. Check your inbox and press Enter to continue.")

        print("Sending mail through the Amazon SES SMTP server.")
        boto3_session = boto3.Session()
        region = boto3_session.region_name
        credentials = boto3_session.get_credentials()
        port = 587
        smtp_server = f'email-smtp.{region}.amazonaws.com'
        password = calculate_key(credentials.secret_key, region)
        message = """
Subject: Hi there

This message is sent from the Amazon SES SMTP mail demo."""
        context = ssl.create_default_context()
        with smtplib.SMTP(smtp_server, port) as server:
            server.starttls(context=context)
            server.login(credentials.access_key, password)
            server.sendmail(email, email, message)
        print("Mail sent. Check your inbox!")

    if ses_template.template is not None:
        print("Deleting demo template.")
        ses_template.delete_template()
    if verified:
        answer = input(f"Do you want to remove {email} from Amazon SES (y/n)? ")
        if answer.lower() == 'y':
            ses_identity.delete_identity(email)
    print("Thanks for watching!")
    print('-'*88)

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateTemplate](#)
  - [DeleteIdentity](#)
  - [DeleteTemplate](#)
  - [GetIdentityVerificationAttributes](#)
  - [GetTemplate](#)
  - [ListIdentities](#)
  - [ListTemplates](#)
  - [SendEmail](#)
  - [SendTemplatedEmail](#)
  - [UpdateTemplate](#)
  - [VerifyDomainIdentity](#)
  - [VerifyEmailIdentity](#)

## Cross-service examples for Amazon SES using AWS SDKs

The following code examples show how to use Amazon Simple Email Service with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Build an Amazon Transcribe streaming app \(p. 1912\)](#)
- [Create a web application to track DynamoDB data \(p. 1913\)](#)
- [Create an Amazon Redshift item tracker \(p. 1914\)](#)
- [Create an Aurora Serverless work item tracker \(p. 1915\)](#)
- [Detect PPE in images with Amazon Rekognition using an AWS SDK \(p. 1918\)](#)
- [Detect objects in images with Amazon Rekognition using an AWS SDK \(p. 1919\)](#)
- [Detect people and objects in a video with Amazon Rekognition using an AWS SDK \(p. 1921\)](#)
- [Use Step Functions to invoke Lambda functions \(p. 1922\)](#)

## Build an Amazon Transcribe streaming app

The following code example shows how to build an app that records, transcribes, and translates live audio in real-time, and emails the results.

JavaScript

### SDK for JavaScript V3

Shows how to use Amazon Transcribe to build an app that records, transcribes, and translates live audio in real-time, and emails the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Comprehend

- Amazon SES
- Amazon Transcribe
- Amazon Translate

## Create a web application to track DynamoDB data

The following code examples show how to create a web application that tracks work items in an Amazon DynamoDB table and uses Amazon Simple Email Service (Amazon SES) to send reports.

.NET

### AWS SDK for .NET

Shows how to use the Amazon DynamoDB .NET API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon SES

Java

### SDK for Java 2.x

Shows how to use the Amazon DynamoDB API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon SES

JavaScript

### SDK for JavaScript V3

Shows how to use the Amazon DynamoDB API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon SES

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to use the Amazon DynamoDB API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Amazon SES

## Python

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) to create a REST service that tracks work items in Amazon DynamoDB and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses the Flask web framework to handle HTTP routing and integrates with a React webpage to present a fully functional web application.

- Build a Flask REST service that integrates with AWS services.
- Read, write, and update work items that are stored in a DynamoDB table.
- Use Amazon SES to send email reports of work items.

For complete source code and instructions on how to set up and run, see the full example in the [AWS Code Examples Repository](#) on GitHub.

#### Services used in this example

- DynamoDB
- Amazon SES

## Create an Amazon Redshift item tracker

The following code examples show how to create a web application that tracks and reports on work items using an Amazon Redshift database.

## Java

### SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon Redshift database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Redshift data and for use by a React application, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Redshift

- Amazon SES

Kotlin

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to create a web application that tracks and reports on work items stored in an Amazon Redshift database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Redshift data and for use by a React application, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Redshift
- Amazon SES

## Create an Aurora Serverless work item tracker

The following code examples show how to create a web application that tracks work items in an Amazon Aurora Serverless database and uses Amazon Simple Email Service (Amazon SES) to send reports.

.NET

#### AWS SDK for .NET

Shows how to use the AWS SDK for .NET to create a web application that tracks work items in an Amazon Aurora database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with a RESTful .NET backend.

- Integrate a React web application with AWS services.
- List, add, update, and delete items in an Aurora table.
- Send an email report of filtered work items using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

C++

#### SDK for C++

Shows how to create a web application that tracks and reports on work items stored in an Amazon Aurora Serverless database.

For complete source code and instructions on how to set up a C++ REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Java

#### SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

For complete source code and instructions on how to set up and run an example that uses the JDBC API, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

JavaScript

#### SDK for JavaScript V3

Shows how to use the AWS SDK for JavaScript (v3) to create a web application that tracks work items in an Amazon Aurora database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with an Express Node.js backend.

- Integrate a React.js web application with AWS services.
- List, add, and update items in an Aurora table.
- Send an email report of filtered work items by using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service

- Amazon SES

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

#### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

PHP

### SDK for PHP

Shows how to use the AWS SDK for PHP to create a web application that tracks work items in an Amazon RDS database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with a RESTful PHP backend.

- Integrate a React.js web application with AWS services.
- List, add, update, and delete items in an Amazon RDS table.
- Send an email report of filtered work items using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Python

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) to create a REST service that tracks work items in an Amazon Aurora Serverless database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses the Flask web framework to handle HTTP routing and integrates with a React webpage to present a fully functional web application.

- Build a Flask REST service that integrates with AWS services.
- Read, write, and update work items that are stored in an Aurora Serverless database.
- Create an AWS Secrets Manager secret that contains database credentials and use it to authenticate calls to the database.
- Use Amazon SES to send email reports of work items.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

## **Detect PPE in images with Amazon Rekognition using an AWS SDK**

The following code examples show how to build an app that uses Amazon Rekognition to detect Personal Protective Equipment (PPE) in images.

Java

#### **SDK for Java 2.x**

Shows how to create an AWS Lambda function that detects images with Personal Protective Equipment.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

#### **SDK for JavaScript V3**

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an application to detect personal protective equipment (PPE) in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app saves the results to an Amazon DynamoDB table, and sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for PPE using Amazon Rekognition.
- Verify an email address for Amazon SES.

- Update a DynamoDB table with results.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

## Detect objects in images with Amazon Rekognition using an AWS SDK

The following code examples show how to build an app that uses Amazon Rekognition to detect objects by category in images.

.NET

#### **AWS SDK for .NET**

Shows how to use Amazon Rekognition .NET API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

#### **SDK for Java 2.x**

Shows how to use Amazon Rekognition Java API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

## JavaScript

### SDK for JavaScript V3

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for objects using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to use Amazon Rekognition Kotlin API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Python

### SDK for Python (Boto3)

Shows you how to use the AWS SDK for Python (Boto3) to create a web application that lets you do the following:

- Upload photos to an Amazon Simple Storage Service (Amazon S3) bucket.
- Use Amazon Rekognition to analyze and label the photos.
- Use Amazon Simple Email Service (Amazon SES) to send email reports of image analysis.

This example contains two main components: a webpage written in JavaScript that is built with React, and a REST service written in Python that is built with Flask-RESTful.

You can use the React webpage to:

- Display a list of images that are stored in your S3 bucket.
- Upload images from your computer to your S3 bucket.
- Display images and labels that identify items that are detected in the image.
- Get a report of all images in your S3 bucket and send an email of the report.

The webpage calls the REST service. The service sends requests to AWS to perform the following actions:

- Get and filter the list of images in your S3 bucket.
- Upload photos to your S3 bucket.
- Use Amazon Rekognition to analyze individual photos and get a list of labels that identify items that are detected in the photo.
- Analyze all photos in your S3 bucket and use Amazon SES to email a report.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### **Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Detect people and objects in a video with Amazon Rekognition using an AWS SDK

The following code examples show how to detect people and objects in a video with Amazon Rekognition.

Java

### **SDK for Java 2.x**

Shows how to use Amazon Rekognition Java API to create an app to detect faces and objects in videos located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### **Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

### **SDK for JavaScript V3**

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an app to detect faces and objects in videos located in an Amazon Simple Storage Service (Amazon S3)

bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for PPE using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Amazon Rekognition
- Amazon S3
- Amazon SES

## **Use Step Functions to invoke Lambda functions**

The following code examples show how to create an AWS Step Functions state machine that invokes AWS Lambda functions in sequence.

Java

#### **SDK for Java 2.x**

Shows how to create an AWS serverless workflow by using AWS Step Functions and the AWS SDK for Java 2.x. Each workflow step is implemented using an AWS Lambda function.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

JavaScript

#### **SDK for JavaScript V3**

Shows how to create an AWS serverless workflow by using AWS Step Functions and the AWS SDK for JavaScript. Each workflow step is implemented using an AWS Lambda function.

Lambda is a compute service that enables you to run code without provisioning or managing servers. Step Functions is a serverless orchestration service that lets you combine Lambda functions and other AWS services to build business-critical applications.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

### Services used in this example

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

## Code examples for Amazon SES API v2 using AWS SDKs

The following code examples show how to use Amazon Simple Email Service API v2 with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon SES API v2 using AWS SDKs \(p. 1923\)](#)
  - Create an Amazon SES API v2 contact in a contact list using an AWS SDK (p. 1923)
  - Create an Amazon SES API v2 contact list using an AWS SDK (p. 1924)
  - Get information about an Amazon SES API v2 identity using an AWS SDK (p. 1924)
  - List the Amazon SES API v2 contact lists using an AWS SDK (p. 1925)
  - List the contacts in an Amazon SES API v2 contact list using an AWS SDK (p. 1926)
  - Send an Amazon SES API v2 email using an AWS SDK (p. 1926)

## Actions for Amazon SES API v2 using AWS SDKs

The following code examples show how to use Amazon Simple Email Service API v2 with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create an Amazon SES API v2 contact in a contact list using an AWS SDK \(p. 1923\)](#)
- [Create an Amazon SES API v2 contact list using an AWS SDK \(p. 1924\)](#)
- [Get information about an Amazon SES API v2 identity using an AWS SDK \(p. 1924\)](#)
- [List the Amazon SES API v2 contact lists using an AWS SDK \(p. 1925\)](#)
- [List the contacts in an Amazon SES API v2 contact list using an AWS SDK \(p. 1926\)](#)
- [Send an Amazon SES API v2 email using an AWS SDK \(p. 1926\)](#)

## Create an Amazon SES API v2 contact in a contact list using an AWS SDK

The following code example shows how to create an Amazon SES API v2 contact in a contact list.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn add_contact(client: &Client, list: &str, email: &str) -> Result<(), Error> {
    client
        .create_contact()
        .contact_list_name(list)
        .email_address(email)
        .send()
        .await?;

    println!("Created contact");

    Ok(())
}
```

- For API details, see [CreateContact](#) in *AWS SDK for Rust API reference*.

## Create an Amazon SES API v2 contact list using an AWS SDK

The following code example shows how to create an Amazon SES API v2 contact list.

Rust

### SDK for Rust

**Note**

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_list(client: &Client, contact_list: &str) -> Result<(), Error> {
    client
        .create_contact_list()
        .contact_list_name(contact_list)
        .send()
        .await?;

    println!("Created contact list.");

    Ok(())
}
```

- For API details, see [CreateContactList](#) in *AWS SDK for Rust API reference*.

## Get information about an Amazon SES API v2 identity using an AWS SDK

The following code example shows how to get Amazon SES API v2 identity information.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Determines whether an email address has been verified.

```
async fn is_verified(client: &Client, email: &str) -> Result<(), Error> {
    let resp = client
        .get_email_identity()
        .email_identity(email)
        .send()
        .await?;

    if resp.verified_for_sending_status() {
        println!("The address is verified");
    } else {
        println!("The address is not verified");
    }

    Ok(())
}
```

- For API details, see [GetEmailIdentity](#) in *AWS SDK for Rust API reference*.

## List the Amazon SES API v2 contact lists using an AWS SDK

The following code example shows how to list the Amazon SES API v2 contact lists.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_lists(client: &Client) -> Result<(), Error> {
    let resp = client.list_contact_lists().send().await?;

    println!("Contact lists:");

    for list in resp.contact_lists().unwrap_or_default() {
        println!("  {}", list.contact_list_name().unwrap_or_default());
    }

    Ok(())
}
```

- For API details, see [ListContactLists](#) in *AWS SDK for Rust API reference*.

## List the contacts in an Amazon SES API v2 contact list using an AWS SDK

The following code example shows how to list the contacts in an Amazon SES API v2 contact list.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_contacts(client: &Client, list: &str) -> Result<(), Error> {
    let resp = client
        .list_contacts()
        .contact_list_name(list)
        .send()
        .await?;

    println!("Contacts:");

    for contact in resp.contacts().unwrap_or_default() {
        println!("  {}", contact.email_address().unwrap_or_default());
    }

    Ok(())
}
```

- For API details, see [ListContacts](#) in *AWS SDK for Rust API reference*.

## Send an Amazon SES API v2 email using an AWS SDK

The following code examples show how to send an Amazon SES API v2 email.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Sends a message.

```
public static void send(SesV2Client client,
                      String sender,
                      String recipient,
                      String subject,
                      String bodyHTML
){
```

```
Destination destination = Destination.builder()
    .toAddresses(recipient)
    .build();

Content content = Content.builder()
    .data(bodyHTML)
    .build();

Content sub = Content.builder()
    .data(subject)
    .build();

Body body = Body.builder()
    .html(content)
    .build();

Message msg = Message.builder()
    .subject(sub)
    .body(body)
    .build();

EmailContent emailContent = EmailContent.builder()
    .simple(msg)
    .build();

SendEmailRequest emailRequest = SendEmailRequest.builder()
    .destination(destination)
    .content(emailContent)
    .fromEmailAddress(sender)
    .build();

try {
    System.out.println("Attempting to send an email through Amazon SES
" + "using the AWS SDK for Java...");
    client.sendEmail(emailRequest);
    System.out.println("email was sent");

} catch (SesV2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [SendEmail](#) in *AWS SDK for Java 2.x API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Sends a message to all members of the contact list.

```
async fn send_message(
    client: &Client,
```

```
list: &str,
from: &str,
subject: &str,
message: &str,
) -> Result<(), Error> {
    // Get list of email addresses from contact list.
    let resp = client
        .list_contacts()
        .contact_list_name(list)
        .send()
        .await?;

    let contacts = resp.contacts().unwrap_or_default();

    let cs: String = contacts
        .iter()
        .map(|i| i.email_address().unwrap_or_default())
        .collect();

    let dest = Destination::builder().to_addresses(cs).build();
    let subject_content =
Content::builder().data(subject).charset("UTF-8").build();
    let body_content = Content::builder().data(message).charset("UTF-8").build();
    let body = Body::builder().text(body_content).build();

    let msg = Message::builder()
        .subject(subject_content)
        .body(body)
        .build();

    let email_content = EmailContent::builder().simple(msg).build();

    client
        .send_email()
        .from_email_address(from)
        .destination(dest)
        .content(email_content)
        .send()
        .await?;

    println!("Email sent to list");
}

Ok(())
}
```

- For API details, see [SendEmail](#) in *AWS SDK for Rust API reference*.

## Code examples for Amazon SNS using AWS SDKs

The following code examples show how to use Amazon Simple Notification Service with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon SNS using AWS SDKs \(p. 1929\)](#)
  - [Add tags to an Amazon SNS topic using an AWS SDK \(p. 1930\)](#)
  - [Check whether a phone number is opted out of Amazon SNS using an AWS SDK \(p. 1931\)](#)
  - [Confirm an endpoint owner wants to receive Amazon SNS messages using an AWS SDK \(p. 1935\)](#)
  - [Create an Amazon SNS topic using an AWS SDK \(p. 1937\)](#)

- Delete an Amazon SNS subscription using an AWS SDK (p. 1943)
- Delete an Amazon SNS topic using an AWS SDK (p. 1947)
- Get the properties of an Amazon SNS topic using an AWS SDK (p. 1952)
- Get the settings for sending Amazon SNS SMS messages using an AWS SDK (p. 1956)
- List phone numbers opted out of Amazon SNS using an AWS SDK (p. 1959)
- List the subscribers of an Amazon SNS topic using an AWS SDK (p. 1961)
- List Amazon SNS topics using an AWS SDK (p. 1967)
- Publish an Amazon SNS SMS text message using an AWS SDK (p. 1973)
- Publish to an Amazon SNS topic using an AWS SDK (p. 1978)
- Set a dead-letter queue for an Amazon SNS subscription using an AWS SDK (p. 1984)
- Set an Amazon SNS filter policy using an AWS SDK (p. 1985)
- Set the default settings for sending Amazon SNS SMS messages using an AWS SDK (p. 1987)
- Set Amazon SNS topic attributes using an AWS SDK (p. 1989)
- Subscribe a Lambda function to receive notifications from an Amazon SNS topic using an AWS SDK (p. 1993)
- Subscribe a mobile application to an Amazon SNS topic using an AWS SDK (p. 1996)
- Subscribe an HTTP endpoint to an Amazon SNS topic using an AWS SDK (p. 1998)
- Subscribe an email address to an Amazon SNS topic using an AWS SDK (p. 2000)
- Scenarios for Amazon SNS using AWS SDKs (p. 2007)
  - Create a platform endpoint for Amazon SNS push notifications using an AWS SDK (p. 2007)
  - Create and publish to a FIFO Amazon SNS topic using an AWS SDK (p. 2008)
  - Publish SMS messages to an Amazon SNS topic using an AWS SDK (p. 2011)
  - Publish a large message to Amazon SNS with Amazon S3 using an AWS SDK (p. 2013)
- Cross-service examples for Amazon SNS using AWS SDKs (p. 2015)
  - Build an application to submit data to a DynamoDB table (p. 2015)
  - Build a publish and subscription application that translates messages (p. 2016)
  - Create an Amazon Textract explorer application (p. 2017)
  - Detect people and objects in a video with Amazon Rekognition using an AWS SDK (p. 2018)
  - Use API Gateway to invoke a Lambda function (p. 2019)
  - Use scheduled events to invoke a Lambda function (p. 2020)

## Actions for Amazon SNS using AWS SDKs

The following code examples show how to use Amazon Simple Notification Service with AWS SDKs. Each example calls an individual service function.

### Examples

- Add tags to an Amazon SNS topic using an AWS SDK (p. 1930)
- Check whether a phone number is opted out of Amazon SNS using an AWS SDK (p. 1931)
- Confirm an endpoint owner wants to receive Amazon SNS messages using an AWS SDK (p. 1935)
- Create an Amazon SNS topic using an AWS SDK (p. 1937)
- Delete an Amazon SNS subscription using an AWS SDK (p. 1943)
- Delete an Amazon SNS topic using an AWS SDK (p. 1947)
- Get the properties of an Amazon SNS topic using an AWS SDK (p. 1952)
- Get the settings for sending Amazon SNS SMS messages using an AWS SDK (p. 1956)

- List phone numbers opted out of Amazon SNS using an AWS SDK (p. 1959)
- List the subscribers of an Amazon SNS topic using an AWS SDK (p. 1961)
- List Amazon SNS topics using an AWS SDK (p. 1967)
- Publish an Amazon SNS SMS text message using an AWS SDK (p. 1973)
- Publish to an Amazon SNS topic using an AWS SDK (p. 1978)
- Set a dead-letter queue for an Amazon SNS subscription using an AWS SDK (p. 1984)
- Set an Amazon SNS filter policy using an AWS SDK (p. 1985)
- Set the default settings for sending Amazon SNS SMS messages using an AWS SDK (p. 1987)
- Set Amazon SNS topic attributes using an AWS SDK (p. 1989)
- Subscribe a Lambda function to receive notifications from an Amazon SNS topic using an AWS SDK (p. 1993)
- Subscribe a mobile application to an Amazon SNS topic using an AWS SDK (p. 1996)
- Subscribe an HTTP endpoint to an Amazon SNS topic using an AWS SDK (p. 1998)
- Subscribe an email address to an Amazon SNS topic using an AWS SDK (p. 2000)

## Add tags to an Amazon SNS topic using an AWS SDK

The following code examples show how to add tags to an Amazon SNS topic.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void addTopicTags(SnsClient snsClient, String topicArn) {  
  
    try {  
        Tag tag = Tag.builder()  
            .key("Team")  
            .value("Development")  
            .build();  
  
        Tag tag2 = Tag.builder()  
            .key("Environment")  
            .value("Gamma")  
            .build();  
  
        List<Tag> tagList = new ArrayList<>();  
        tagList.add(tag);  
        tagList.add(tag2);  
  
        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()  
            .resourceArn(topicArn)  
            .tags(tagList)  
            .build();  
  
        snsClient.tagResource(tagResourceRequest);  
        System.out.println("Tags have been added to "+topicArn);  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [TagResource](#) in *AWS SDK for Java 2.x API Reference*.

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun addTopicTags(topicArn: String) {

    val tag = Tag {
        key = "Team"
        value = "Development"
    }

    val tag2 = Tag {
        key = "Environment"
        value = "Gamma"
    }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

    val request = TagResourceRequest {
        resourceArn = topicArn
        tags = tagList
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- For API details, see [TagResource](#) in *AWS SDK for Kotlin API reference*.

## Check whether a phone number is opted out of Amazon SNS using an AWS SDK

The following code examples show how to check whether a phone number is opted out of receiving Amazon SNS messages.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example shows how to use the Amazon Simple Notification Service
/// (Amazon SNS) to check whether a phone number has been opted out. The
/// example was created using the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
/// </summary>
public class IsPhoneNumOptedOut
{
    public static async Task Main()
    {
        string phoneNumber = "+15551112222";

        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        await CheckIfOptedOutAsync(client, phoneNumber);
    }

    /// <summary>
    /// Checks to see if the supplied phone number has been opted out.
    /// </summary>
    /// <param name="client">The initialized Amazon SNS Client object used
    /// to check if the phone number has been opted out.</param>
    /// <param name="phoneNumber">A string representing the phone number
    /// to check.</param>
    public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string phoneNumber)
    {
        var request = new CheckIfPhoneNumberIsOptedOutRequest
        {
            PhoneNumber = phoneNumber,
        };

        try
        {
            var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                string optOutStatus = response.IsOptedOut ? "opted out" : "not
opted out.";
                Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
            }
        }
        catch (AuthorizationErrorException ex)
        {
            Console.WriteLine($"{ex.Message}");
        }
    }
}
```

- For API details, see [CheckIfPhoneNumberIsOptedOut](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void checkPhone(SnsClient snsClient, String phoneNumber) {  
  
    try {  
        CheckIfPhoneNumberIsOptedOutRequest request =  
CheckIfPhoneNumberIsOptedOutRequest.builder()  
            .phoneNumber(phoneNumber)  
            .build();  
  
        CheckIfPhoneNumberIsOptedOutResponse result =  
snsClient.checkIfPhoneNumberIsOptedOut(request);  
        System.out.println(result.isOptedOut() + "Phone Number " + phoneNumber  
+ " has Opted Out of receiving sns messages." +  
        "\n\nStatus was " + result.sdkHttpResponse().statusCode());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [CheckIfPhoneNumberIsOptedOut](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create SNS service object.  
const snsClient = new SNSClient({ region: REGION });  
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js  
import {CheckIfPhoneNumberIsOptedOutCommand } from "@aws-sdk/client-sns";  
import {snsClient } from "./libs/snsClient.js";  
  
// Set the parameters  
const params = { phoneNumber: "353861230764" }; //PHONE_NUMBER, in the E.164 phone  
number structure  
  
const run = async () => {
```

```
try {
    const data = await snsClient.send(
        new CheckIfPhoneNumberIsOptedOutCommand(params)
    );
    console.log("Success.", data);
    return data; // For unit tests.
} catch (err) {
    console.log("Error", err.stack);
}
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CheckIfPhoneNumberIsOptedOut](#) in [AWS SDK for JavaScript API Reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Indicates whether the phone number owner has opted out of receiving SMS messages
 * from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

$SnsClient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
    $result = $SnsClient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [CheckIfPhoneNumberIsOptedOut](#) in [AWS SDK for PHP API Reference](#).

## Confirm an endpoint owner wants to receive Amazon SNS messages using an AWS SDK

The following code examples show how to confirm the owner of an endpoint wants to receive Amazon SNS messages by validating the token sent to the endpoint by an earlier Subscribe action.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void confirmSub(SnsClient snsClient, String subscriptionToken,  
String topicArn ) {  
  
    try {  
        ConfirmSubscriptionRequest request =  
ConfirmSubscriptionRequest.builder()  
            .token(subscriptionToken)  
            .topicArn(topicArn)  
            .build();  
  
        ConfirmSubscriptionResponse result =  
snsClient.confirmSubscription(request);  
        System.out.println("\n\nStatus was " +  
result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n" +  
result.subscriptionArn());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ConfirmSubscription](#) in *AWS SDK for Java 2.x API Reference*.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create SNS service object.  
const snsClient = new SNSClient({ region: REGION });  
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {ConfirmSubscriptionCommand } from "@aws-sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const params = {
  Token: "TOKEN", // Required. Token sent to the endpoint by an earlier Subscribe
  action. */
  TopicArn: "TOPIC_ARN", // Required
  AuthenticateOnUnsubscribe: "true", // 'true' or 'false'
};

const run = async () => {
  try {
    const data = await snsClient.send(new ConfirmSubscriptionCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ConfirmSubscription](#) in [AWS SDK for JavaScript API Reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Verifies an endpoint owner's intent to receive messages by validating the token
 * sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
```

```
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For API details, see [ConfirmSubscription](#) in *AWS SDK for PHP API Reference*.

## Create an Amazon SNS topic using an AWS SDK

The following code examples show how to create an Amazon SNS topic.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example shows how to use Amazon Simple Notification Service
/// (Amazon SNS) to add a new Amazon SNS topic. The example was created
/// using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class CreateSNSTopic
{
    public static async Task Main()
    {
        string topicName = "ExampleSNSTopic";

        IAmazonSimpleNotificationService client = new
        AmazonSimpleNotificationServiceClient();

        var topicArn = await CreateSNSTopicAsync(client, topicName);
        Console.WriteLine($"New topic ARN: {topicArn}");
    }

    /// <summary>
    /// Creates a new SNS topic using the supplied topic name.
    /// </summary>
    /// <param name="client">The initialized SNS client object used to
    /// create the new topic.</param>
    /// <param name="topicName">A string representing the topic name.</param>
    /// <returns>The Amazon Resource Name (ARN) of the created topic.</returns>
    public static async Task<string>
CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
    {
        var request = new CreateTopicRequest
        {
            Name = topicName,
        };

        var response = await client.CreateTopicAsync(request);
```

```
        return response.TopicArn;
    }

}
```

- For API details, see [CreateTopic in AWS SDK for .NET API Reference](#).

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::String topic_name = argv[1];
    Aws::SNS::SNSClient sns;

    Aws::SNS::Model::CreateTopicRequest ct_req;
    ct_req.SetName(topic_name);

    auto ct_out = sns.CreateTopic(ct_req);

    if (ct_out.IsSuccess())
    {
        std::cout << "Successfully created topic " << topic_name << std::endl;
    }
    else
    {
        std::cout << "Error creating topic " << topic_name << ":" <<
            ct_out.GetError().GetMessage() << std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [CreateTopic in AWS SDK for C++ API Reference](#).

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

- For API details, see [CreateTopic in AWS SDK for Go API Reference](#).

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createSNSTopic(SnsClient snsClient, String topicName ) {  
  
    CreateTopicResponse result = null;  
    try {  
        CreateTopicRequest request = CreateTopicRequest.builder()  
            .name(topicName)  
            .build();  
  
        result = snsClient.createTopic(request);  
        return result.topicArn();  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [CreateTopic in AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create SNS service object.  
const snsClient = new SNSClient({ region: REGION });  
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js  
import {CreateTopicCommand} from "@aws-sdk/client-sns";  
import {snsClient} from "./libs/snsClient.js";  
  
// Set the parameters  
const params = { Name: "TOPIC_NAME" }; //TOPIC_NAME  
  
const run = async () => {  
    try {  
        const data = await snsClient.send(new CreateTopicCommand(params));  
        console.log("Success.", data);  
        return data; // For unit tests.  
    } catch (err) {  
        console.log("Error", err.stack);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [CreateTopic](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createSNSTopic(topicName: String): String {  
  
    val request = CreateTopicRequest {  
        name = topicName  
    }  
  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        val result = snsClient.createTopic(request)  
        return result.topicArn.toString()  
    }  
}
```

- For API details, see [CreateTopic](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';  
  
use Aws\Sns\SnsClient;  
use Aws\Exception\AwsException;  
  
/**  
 * Create a Simple Notification Service topics in your AWS account at the requested  
region.  
 *  
 * This code expects that you have AWS credentials set up per:  
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/  
guide_credentials.html  
 */  
  
$SnSclient = new SnsClient([  
    'profile' => 'default',  
    'region' => 'us-east-1',  
    'version' => '2010-03-31'  
]);  
  
$topicname = 'myTopic';  
  
try {  
    $result = $SnSclient->createTopic([
```

```
        'Name' => $topicname,
    ]);
var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [CreateTopic](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_topic(self, name):
        """
        Creates a notification topic.

        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
        else:
            return topic
```

- For API details, see [CreateTopic](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-sns" # v2: require 'aws-sdk'

def topic_created?(sns_client, topic_name)
```

```
    sns_client.create_topic(name: topic_name)
    rescue StandardError => e
      puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    end

    # Full example call:
def run_me
  topic_name = "TOPIC_NAME"
  region = "REGION"

  sns_client = Aws::SNS::Client.new(region: region)

  puts "Creating the topic '#{topic_name}'..."

  if topic_created?(sns_client, topic_name)
    puts "The topic was created."
  else
    puts "The topic was not created. Stopping program."
    exit 1
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [CreateTopic in AWS SDK for Ruby API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- For API details, see [CreateTopic in AWS SDK for Rust API reference](#).

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result is  
    returned for testing purpose "  
    MESSAGE 'SNS topic created' TYPE 'I'.  
    CATCH /aws1/cx_snstopiclimitexcdex.  
    MESSAGE 'Unable to create more topics as you have reached the maximum  
    number of topics allowed.' TYPE 'E'.  
    ENDTRY.
```

- For API details, see [CreateTopic in AWS SDK for SAP ABAP API reference](#).

## Delete an Amazon SNS subscription using an AWS SDK

The following code examples show how to delete an Amazon SNS subscription.

### .NET

#### AWS SDK for .NET

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>  
/// Given the ARN for an Amazon SNS subscription, this method deletes  
/// the subscription.  
///</summary>  
///<param name="client">The initialized Amazon SNS client object, used  
/// to delete an Amazon SNS subscription.</param>  
///<param name="subscriptionArn">The ARN of the subscription to delete.</param>  
public static async Task TopicUnsubscribeAsync(  
    IAmazonSimpleNotificationService client,  
    string subscriptionArn)  
{  
    var response = await client.UnsubscribeAsync(subscriptionArn);  
}
```

- For API details, see [Unsubscribe in AWS SDK for .NET API Reference](#).

### C++

#### SDK for C++

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;  
Aws::InitAPI(options);
```

```
{  
    Aws::SNS::SNSClient sns;  
    Aws::String subscription_arn = argv[1];  
  
    Aws::SNS::Model::UnsubscribeRequest s_req;  
    s_req.SetSubscriptionArn(subscription_arn);  
  
    auto s_out = sns.Unsubscribe(s_req);  
  
    if (s_out.IsSuccess())  
    {  
        std::cout << "Unsubscribed successfully " << std::endl;  
    }  
    else  
    {  
        std::cout << "Error while unsubscribing " << s_out.GetError().GetMessage()  
            << std::endl;  
    }  
}  
  
Aws::ShutdownAPI(options);
```

- For API details, see [Unsubscribe in AWS SDK for C++ API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void unSub(SnsClient snsClient, String subscriptionArn) {  
  
    try {  
        UnsubscribeRequest request = UnsubscribeRequest.builder()  
            .subscriptionArn(subscriptionArn)  
            .build();  
  
        UnsubscribeResponse result = snsClient.unsubscribe(request);  
        System.out.println("\n\nStatus was " +  
            result.sdkHttpResponse().statusCode()  
            + "\n\nSubscription was removed for " + request.subscriptionArn());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [Unsubscribe in AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {UnsubscribeCommand } from "@aws-sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const params = { SubscriptionArn: "TOPIC_SUBSCRIPTION_ARN" }; // 
TOPIC_SUBSCRIPTION_ARN

const run = async () => {
  try {
    const data = await snsClient.send(new UnsubscribeCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Unsubscribe in AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun unSub(subscriptionArnVal: String) {

    val request = UnsubscribeRequest {
        subscriptionArn = subscriptionArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- For API details, see [Unsubscribe in AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [Unsubscribe](#) in [AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def delete_subscription(subscription):
        """
```

```
Unsubscribes and deletes a subscription.  
"""  
try:  
    subscription.delete()  
    logger.info("Deleted subscription %s.", subscription.arn)  
except ClientError:  
    logger.exception("Couldn't delete subscription %s.", subscription.arn)  
    raise
```

- For API details, see [Unsubscribe in AWS SDK for Python \(Boto3\) API Reference](#).

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).  
    MESSAGE 'Subscription Deleted' TYPE 'I'.  
    CATCH /aws1/cx_snsnotfoundexception.  
        MESSAGE 'Subscription does not exist' TYPE 'E'.  
        CATCH /aws1/cx_snsinvalidparameterex.  
            MESSAGE 'Subscription with "PendingConfirmation" status cannot be  
            deleted/unsubscribed, subscription should be confirmed before perform unsubscribe  
            operation' TYPE 'E'.  
        ENDTRY.
```

- For API details, see [Unsubscribe in AWS SDK for SAP ABAP API reference](#).

## Delete an Amazon SNS topic using an AWS SDK

The following code examples show how to delete an Amazon SNS topic and all subscriptions to that topic.

### .NET

#### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Threading.Tasks;  
using Amazon.SimpleNotificationService;  
  
/// <summary>  
/// This example deletes an existing Amazon Simple Notification Service  
/// (Amazon SNS) topic. The example was created using the AWS SDK for .NET  
/// version 3.7 and .NET Core 5.0.  
/// </summary>
```

```
public class DeleteSNSTopic
{
    public static async Task Main()
    {
        string topicArn = "arn:aws:sns:us-east-2:012345678901:ExampleSNSTopic";
        IAmazonSimpleNotificationService client = new
        AmazonSimpleNotificationServiceClient();

        var response = await client.DeleteTopicAsync(topicArn);
    }
}
```

- For API details, see [DeleteTopic in AWS SDK for .NET API Reference](#).

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::String topic_arn = argv[1];
    Aws::SNS::SNSClient sns;

    Aws::SNS::Model::DeleteTopicRequest dt_req;
    dt_req.SetTopicArn(topic_arn);

    auto dt_out = sns.DeleteTopic(dt_req);

    if (dt_out.IsSuccess())
    {
        std::cout << "Successfully deleted topic " << topic_arn << std::endl;
    }
    else
    {
        std::cout << "Error deleting topic " << topic_arn << ":" <<
            dt_out.GetError().GetMessage() << std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [DeleteTopic in AWS SDK for C++ API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
```

```
try {
    DeleteTopicRequest request = DeleteTopicRequest.builder()
        .topicArn(topicArn)
        .build();

    DeleteTopicResponse result = snsClient.deleteTopic(request);
    System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [DeleteTopic in AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Load the AWS SDK for Node.js

// Import required AWS SDK clients and commands for Node.js
import {DeleteTopicCommand} from "@aws-sdk/client-sns";
import {snsClient} from "./libs/snsClient.js";

// Set the parameters
const params = { TopicArn: "TOPIC_ARN" }; //TOPIC_ARN

const run = async () => {
    try {
        const data = await snsClient.send(new DeleteTopicCommand(params));
        console.log("Success.", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err.stack);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteTopic in AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteSNSTopic(topicArnVal: String) {  
  
    val request = DeleteTopicRequest {  
        topicArn = topicArnVal  
    }  
  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        snsClient.deleteTopic(request)  
        println("$topicArnVal was successfully deleted.")  
    }  
}
```

- For API details, see [DeleteTopic](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';  
  
use Aws\Sns\SnsClient;  
use Aws\Exception\AwsException;  
  
/**  
 * Deletes a SNS topic and all its subscriptions.  
 *  
 * This code expects that you have AWS credentials set up per:  
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/  
 * guide_credentials.html  
 */  
  
$SnSclient = new SnsClient([  
    'profile' => 'default',  
    'region' => 'us-east-1',  
    'version' => '2010-03-31'  
]);  
  
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';  
  
try {  
    $result = $SnSclient->deleteTopic([  
        'TopicArn' => $topic,  
    ]);  
    var_dump($result);  
}
```

```
    } catch (AwsException $e) {
        // output error message if fails
        error_log($e->getMessage());
    }
}
```

- For API details, see [DeleteTopic in AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- For API details, see [DeleteTopic in AWS SDK for Python \(Boto3\) API Reference](#).

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
    MESSAGE 'SNS topic deleted' TYPE 'I'.
    CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist' TYPE 'E'.
ENDTRY.
```

- For API details, see [DeleteTopic in AWS SDK for SAP ABAP API reference](#).

## Get the properties of an Amazon SNS topic using an AWS SDK

The following code examples show how to get the properties of an Amazon SNS topic.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example shows how to retrieve the attributes of an Amazon Simple
/// Notification Service (Amazon SNS) topic. The example was written using
/// the AWS SDK for .NET 3.7 and .NET Core 5.0.
/// </summary>
public class GetTopicAttributes
{
    public static async Task Main()
    {
        string topicArn = "arn:aws:sns:us-west-2:000000000000:ExampleSNSTopic";
        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        var attributes = await GetTopicAttributesAsync(client, topicArn);
        DisplayTopicAttributes(attributes);
    }

    /// <summary>
    /// Given the ARN of the Amazon SNS topic, this method retrieves the topic
    /// attributes.
    /// </summary>
    /// <param name="client">The initialized Amazon SNS client object used
    /// to retrieve the attributes for the Amazon SNS topic.</param>
    /// <param name="topicArn">The ARN of the topic for which to retrieve
    /// the attributes.</param>
    /// <returns>A Dictionary of topic attributes.</returns>
    public static async Task<Dictionary<string, string>>
GetTopicAttributesAsync(
    IAmazonSimpleNotificationService client,
    string topicArn)
{
    var response = await client.GetTopicAttributesAsync(topicArn);

    return response.Attributes;
}

    /// <summary>
    /// This method displays the attributes for an Amazon SNS topic.
    /// </summary>
    /// <param name="topicAttributes">A Dictionary containing the
    /// attributes for an Amazon SNS topic.</param>
    public static void DisplayTopicAttributes(Dictionary<string, string>
topicAttributes)
{
    foreach (KeyValuePair<string, string> entry in topicAttributes)
{
```

```
        Console.WriteLine($"{entry.Key}: {entry.Value}\n");
    }
}
```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;
    Aws::String topic_arn = argv[1];

    Aws::SNS::Model::GetTopicAttributesRequest gta_req;
    gta_req.SetTopicArn(topic_arn);

    auto gta_out = sns.GetTopicAttributes(gta_req);

    if (gta_out.IsSuccess())
    {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute : gta_out.GetResult().GetAttributes())
        {
            std::cout << "  * " << attribute.first << " : " << attribute.second <<
std::endl;
        }
    }
    else
    {
        std::cout << "Error while getting Topic attributes " <<
gta_out.GetError().GetMessage()
        << std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getSNSTopicAttributes(SnsClient snsClient, String topicArn )
```

```
try {
    GetTopicAttributesRequest request = GetTopicAttributesRequest.builder()
        .topicArn(topicArn)
        .build();

    GetTopicAttributesResponse result =
        snsClient.getTopicAttributes(request);
    System.out.println("\n\nStatus is " +
        result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n" +
        result.attributes());

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {GetTopicAttributesCommand} from "@aws-sdk/client-sns";
import {snsClient} from "./libs/snsClient.js";

// Set the parameters
const params = { TopicArn: "TOPIC_ARN" }; // TOPIC_ARN

const run = async () => {
    try {
        const data = await snsClient.send(new GetTopicAttributesCommand(params));
        console.log("Success.", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err.stack);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [GetTopicAttributes](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Import the SDK and client modules and call the API.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set region
AWS.config.update({region: 'REGION'});

// Create promise and SNS service object
var getTopicAttribsPromise = new AWS.SNS({apiVersion:
  '2010-03-31'}).getTopicAttributes({TopicArn: 'TOPIC_ARN'}).promise();

// Handle promise's fulfilled/rejected states
getTopicAttribsPromise.then(
  function(data) {
    console.log(data);
  }).catch(
  function(err) {
    console.error(err, err.stack);
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetTopicAttributes](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {

    val request = GetTopicAttributesRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for PHP API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_sns->gettopicattributes( iv_topicarn = iv_topic_arn ). "
oo_result is returned for testing purpose "
    DATA(lt_attributes) = oo_result->get_attributes( ).
    MESSAGE 'Retrieved attributes/properties of a topic' TYPE 'I'.
CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'topic does NOT exist' TYPE 'E'.
ENDTRY.
```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for SAP ABAP API reference*.

## Get the settings for sending Amazon SNS SMS messages using an AWS SDK

The following code examples show how to get the settings for sending Amazon SNS SMS messages.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;

    Aws::SNS::Model::GetSMSAttributesRequest gsmst_req;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    gsmst_req.AddAttributes("DefaultSMSType");

    auto gsmst_out = sns.GetSMSAttributes(gsmst_req);

    if (gsmst_out.IsSuccess())
    {
        for (auto const& att : gsmst_out.GetResult().GetAttributes())
        {
            std::cout << att.first << ":" << att.second << std::endl;
        }
    }
    else
    {
        std::cout << "Error while getting SMS Type: '" <<
        gsmst_out.GetError().GetMessage()
        << "'" << std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [GetSMSAttributes](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getSNSAttributes(SnsClient snsClient, String topicArn ) {

    try {
        GetSubscriptionAttributesRequest request =
GetSubscriptionAttributesRequest.builder()
        .subscriptionArn(topicArn)
        .build();

        // Get the Subscription attributes
        GetSubscriptionAttributesResponse res =
snsClient.getSubscriptionAttributes(request);
        Map<String, String> map = res.attributes();
```

```
// Iterate through the map
Iterator iter = map.entrySet().iterator();
while (iter.hasNext()) {
    Map.Entry entry = (Map.Entry) iter.next();
    System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
entry.getValue());
}

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("\n\nStatus was good");
}
```

- For API details, see [GetSMSAttributes](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {GetSMSAttributesCommand} from "@aws-sdk/client-sns";
import {snsClient} from "./libs/snsClient.js";

// Set the parameters
var params = {
    attributes: [
        "DefaultSMSType",
        "ATTRIBUTE_NAME",
        /* more items */
    ],
};

const run = async () => {
    try {
        const data = await snsClient.send(new GetSMSAttributesCommand(params));
        console.log("Success.", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err.stack);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetSMSAttributes](#) in [AWS SDK for JavaScript API Reference](#).

PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Get the type of SMS Message sent by default from the AWS SNS service.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [GetSMSAttributes](#) in [AWS SDK for PHP API Reference](#).

## List phone numbers opted out of Amazon SNS using an AWS SDK

The following code examples show how to list phone numbers that are opted out of receiving Amazon SNS messages.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void list0pts( SnsClient snsClient ) {

    try {
        ListPhoneNumbersOptedOutRequest request =
ListPhoneNumbersOptedOutRequest.builder().build();
        ListPhoneNumbersOptedOutResponse result =
snsClient.listPhoneNumbersOptedOut(request);
        System.out.println("Status is " + result.sdkHttpResponse().statusCode()
+ "\n\nPhone Numbers: \n\n" + result.phoneNumbers());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListPhoneNumbersOptedOut](#) in *AWS SDK for Java 2.x API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 * from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut([
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [ListPhoneNumbersOptedOut](#) in *AWS SDK for PHP API Reference*.

## List the subscribers of an Amazon SNS topic using an AWS SDK

The following code examples show how to retrieve the list of subscribers of an Amazon SNS topic.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example will retrieve a list of the existing Amazon Simple
/// Notification Service (Amazon SNS) subscriptions. The example was
/// created using the AWS SDK for .NET 3.7 and .NET Core 5.0.
/// </summary>
public class ListSubscriptions
{
    public static async Task Main()
    {
        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        var subscriptions = await GetSubscriptionsListAsync(client);

        DisplaySubscriptionList(subscriptions);
    }

    /// <summary>
    /// Gets a list of the existing Amazon SNS subscriptions.
    /// </summary>
    /// <param name="client">The initialized Amazon SNS client object used
    /// to obtain the list of subscriptions.</param>
    /// <returns>A List containing information about each subscription.</returns>
    public static async Task<List<Subscription>>
GetSubscriptionsListAsync(IAmazonSimpleNotificationService client)
    {
        var response = await client.ListSubscriptionsAsync();

        return response.Subscriptions;
    }

    /// <summary>
    /// Display a list of Amazon SNS subscription information.
    /// </summary>
    /// <param name="subscriptionList">A list containing details for existing
    /// Amazon SNS subscriptions.</param>
    public static void DisplaySubscriptionList(List<Subscription>
subscriptionList)
    {
        foreach (var subscription in subscriptionList)
        {
            Console.WriteLine($"Owner: {subscription.Owner}");
            Console.WriteLine($"Subscription ARN:
{subscription.SubscriptionArn}");
        }
    }
}
```

```
        Console.WriteLine($"Topic ARN: {subscription.TopicArn}");
        Console.WriteLine($"Endpoint: {subscription.Endpoint}");
        Console.WriteLine($"Protocol: {subscription.Protocol}");
        Console.WriteLine();
    }
}
```

- For API details, see [ListSubscriptions](#) in *AWS SDK for .NET API Reference*.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;

    Aws::SNS::Model::ListSubscriptionsRequest ls_req;

    auto ls_out = sns.ListSubscriptions(ls_req);

    if (ls_out.IsSuccess())
    {
        std::cout << "Subscriptions list:" << std::endl;
        for (auto const& subscription : ls_out.GetResult().GetSubscriptions())
        {
            std::cout << "  * " << subscription.GetSubscriptionArn() << std::endl;
        }
    }
    else
    {
        std::cout << "Error listing subscriptions " << ls_out.GetError().GetMessage()
        << std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [ListSubscriptions](#) in *AWS SDK for C++ API Reference*.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

- For API details, see [ListSubscriptions](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listSNSSubscriptions( SnsClient snsClient ) {  
  
    try {  
        ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()  
            .build();  
  
        ListSubscriptionsResponse result =  
snsClient.listSubscriptions(request);  
        System.out.println(result.subscriptions());  
  
    } catch (SnsException e) {  
  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListSubscriptions](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create SNS service object.  
const snsClient = new SNSClient({ region: REGION });  
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js  
import {ListSubscriptionsByTopicCommand} from "@aws-sdk/client-sns";  
import {snsClient} from "./libs/snsClient.js";  
  
// Set the parameters  
const params = { TopicArn: "TOPIC_ARN" }; //TOPIC_ARN  
  
const run = async () => {  
    try {  
        const data = await snsClient.send(new ListSubscriptionsByTopicCommand(params));  
        console.log("Success.", data);  
        return data; // For unit tests.  
    } catch (err) {  
        console.log("Error", err.stack);  
    }  
};
```

```
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListSubscriptions](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listSNSSubscriptions() {  
  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})  
        response.subscriptions?.forEach { sub ->  
            println("Sub ARN is ${sub.subscriptionArn}")  
            println("Sub protocol is ${sub.protocol}")  
        }  
    }  
}
```

- For API details, see [ListSubscriptions](#) in [AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';  
  
use Aws\Sns\SnsClient;  
use Aws\Exception\AwsException;  
  
/**  
 * Returns a list of Amazon SNS subscriptions in the requested region.  
 *  
 * This code expects that you have AWS credentials set up per:  
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/  
 * guide_credentials.html  
 */  
  
$SnSclient = new SnsClient([  
    'profile' => 'default',  
    'region' => 'us-east-1',  
    'version' => '2010-03-31'  
]);
```

```
try {
    $result = $SnSclient->listSubscriptions([
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For API details, see [ListSubscriptions](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def list_subscriptions(self, topic=None):
        """
        Lists subscriptions for the current account, optionally limited to a
        specific topic.

        :param topic: When specified, only subscriptions to this topic are
        returned.
        :return: An iterator that yields the subscriptions.
        """
        try:
            if topic is None:
                subs_iter = self.sns_resource.subscriptions.all()
            else:
                subs_iter = topic.subscriptions.all()
                logger.info("Got subscriptions.")
        except ClientError:
            logger.exception("Couldn't get subscriptions.")
            raise
        else:
            return subs_iter
```

- For API details, see [ListSubscriptions](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-sns" # v2: require 'aws-sdk'

def show_subscriptions?(sns_client, topic_arn)
  topic = sns_client.topic(topic_arn)
  topic.subscriptions.each do |s|
    puts s.attributes["Endpoint"]
  end

rescue StandardError => e
  puts "Error while sending the message: #{e.message}"
end

def run_me

  topic_arn = "SNS_TOPIC_ARN"
  region = "REGION"

  sns_client = Aws::SNS::Resource.new(region: region)

  puts "Listing subscriptions to the topic."

  if show_subscriptions?(sns_client, topic_arn)
  else
    puts "There was an error. Stopping program."
    exit 1
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [ListSubscriptions](#) in [AWS SDK for Ruby API Reference](#).

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  oo_result = lo_sns->listsubscriptions( ).                      " oo_result is
  returned for testing purpose "
  DATA(lt_subscriptions) = oo_result->get_subscriptions( ).
  MESSAGE 'Retrieved list of subscriber(s)' TYPE 'I'.
CATCH /aws1/cx_rt_generic.
  MESSAGE 'Unable to list subscriber(s)' TYPE 'E'.
ENDTRY.
```

- For API details, see [ListSubscriptions](#) in [AWS SDK for SAP ABAP API reference](#).

## List Amazon SNS topics using an AWS SDK

The following code examples show how to list Amazon SNS topics.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// An example to list the Amazon Simple Notification Service (Amazon SNS)
/// topics for the default user account. The code was written using the
/// AWS SDK for .NET 3.7 and .NET Core 5.0.
/// </summary>
public class ListSNSTopics
{
    public static async Task Main()
    {
        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        await GetTopicListAsync(client);
    }

    /// <summary>
    /// Retrieves the list of Amazon SNS topics in groups of up to 100
    /// topics.
    /// </summary>
    /// <param name="client">The initialized Amazon SNS client object used
    /// to retrieve the list of topics.</param>
    public static async Task GetTopicListAsync(IAmazonSimpleNotificationService
client)
    {
        // If there are more than 100 Amazon SNS topics, the call to
        // ListTopicsAsync will return a value to pass to the
        // method to retrieve the next 100 (or less) topics.
        string nextToken = string.Empty;

        do
        {
            var response = await client.ListTopicsAsync(nextToken);
            DisplayTopicsList(response.Topics);
            nextToken = response.NextToken;
        }
        while (!string.IsNullOrEmpty(nextToken));
    }

    /// <summary>
    /// Displays the list of Amazon SNS Topic ARNs.
    /// </summary>
    /// <param name="topicList">The list of Topic ARNs.</param>
    public static void DisplayTopicsList(List<Topic> topicList)
    {
        foreach (var topic in topicList)
```

```
        {
            Console.WriteLine($"{topic.TopicArn}");
        }
    }
}
```

- For API details, see [ListTopics](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;

    Aws::SNS::Model::ListTopicsRequest lt_req;

    auto lt_out = sns.ListTopics(lt_req);

    if (lt_out.IsSuccess())
    {
        std::cout << "Topics list:" << std::endl;
        for (auto const &topic : lt_out.GetResult().GetTopics())
        {
            std::cout << "  * " << topic.GetTopicArn() << std::endl;
        }
    }
    else
    {
        std::cout << "Error listing topics " << lt_out.GetError().GetMessage() <<
                     std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [ListTopics](#) in *AWS SDK for C++ API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

- For API details, see [ListTopics](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listSNSTopics(SnsClient snsClient) {  
  
    try {  
        ListTopicsRequest request = ListTopicsRequest.builder()  
            .build();  
  
        ListTopicsResponse result = snsClient.listTopics(request);  
        System.out.println("Status was " +  
            result.sdkHttpResponse().statusCode() + "\n\nTopics\n\n" + result.topics());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListTopics](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create SNS service object.  
const snsClient = new SNSClient({ region: REGION });  
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js  
import {ListTopicsCommand} from "@aws-sdk/client-sns";  
import {snsClient} from "./libs/snsClient.js";  
  
const run = async () => {  
    try {  
        const data = await snsClient.send(new ListTopicsCommand({}));  
        console.log("Success.", data);  
        return data; // For unit tests.  
    } catch (err) {  
        console.log("Error", err.stack);  
    }  
};
```

```
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListTopics](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listSNSTopics() {  
  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        val response = snsClient.listTopics(ListTopicsRequest { })  
        response.topics?.forEach { topic ->  
            println("The topic ARN is ${topic.topicArn}")  
        }  
    }  
}
```

- For API details, see [ListTopics](#) in [AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';  
  
use Aws\Sns\SnsClient;  
use Aws\Exception\AwsException;  
  
/**  
 * Returns a list of the requester's topics from your AWS SNS account in the region  
 * specified.  
 *  
 * This code expects that you have AWS credentials set up per:  
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/  
 * guide_credentials.html  
 */  
  
$SnSclient = new SnsClient([  
    'profile' => 'default',  
    'region' => 'us-east-1',  
    'version' => '2010-03-31'  
]);
```

```
try {
    $result = $SnSclient->listTopics([
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For API details, see [ListTopics](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def list_topics(self):
        """
        Lists topics for the current account.

        :return: An iterator that yields the topics.
        """
        try:
            topics_iter = self.sns_resource.topics.all()
            logger.info("Got topics.")
        except ClientError:
            logger.exception("Couldn't get topics.")
            raise
        else:
            return topics_iter
```

- For API details, see [ListTopics](#) in *AWS SDK for Python (Boto3) API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-sns" # v2: require 'aws-sdk'

def list_topics?(sns_client)
  sns_client.topics.each do |topic|
```

```
    puts topic.arn
rescue StandardError => e
    puts "Error while listing the topics: #{e.message}"
    end
end

def run_me

    region = "REGION"
    sns_client = Aws::SNS::Resource.new(region: region)

    puts "Listing the topics."

    if list_topics?(sns_client)
    else
        puts "The bucket was not created. Stopping program."
        exit 1
    end
end
run_me if $PROGRAM_NAME == __FILE__
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [ListTopics](#) in *AWS SDK for Ruby API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_topics(client: &Client) -> Result<(), Error> {
    let resp = client.list_topics().send().await?;

    println!("Topic ARNs:");

    for topic in resp.topics().unwrap_or_default() {
        println!("{} {}", topic.topic_arn().unwrap_or_default());
    }

    Ok(())
}
```

- For API details, see [ListTopics](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sns->listtopics( ).           " oo_result is returned for  
testing purpose "  
    DATA(lt_topics) = oo_result->get_topics( ).  
    MESSAGE 'Retrieved list of topic(s)' TYPE 'I'.  
CATCH /aws1/cx_rt_generic.  
    MESSAGE 'Unable to list topic(s)' TYPE 'E'.  
ENDTRY.
```

- For API details, see [ListTopics](#) in *AWS SDK for SAP ABAP API reference*.

## Publish an Amazon SNS SMS text message using an AWS SDK

The following code examples show how to publish SMS messages using Amazon SNS.

### .NET

#### AWS SDK for .NET

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon;  
using Amazon.SimpleNotificationService;  
using Amazon.SimpleNotificationService.Model;  
using System;  
using System.Threading.Tasks;  
  
namespace SNSMessageExample  
{  
    class SNSMessage  
    {  
        private AmazonSimpleNotificationServiceClient snsClient;  
  
        /// <summary>  
        /// Constructs a new SNSMessage object initializing the Amazon Simple  
        /// Notification Service (Amazon SNS) client using the supplied  
        /// Region endpoint.  
        /// </summary>  
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in  
        /// sending test messages with this object.</param>  
        public SNSMessage(RegionEndpoint regionEndpoint)  
        {  
            snsClient = new AmazonSimpleNotificationServiceClient(regionEndpoint);  
        }  
  
        /// <summary>  
        /// Sends the SMS message passed in the text parameter to the phone number  
        /// in phoneNum.  
        /// </summary>  
        /// <param name="phoneNum">The ten-digit phone number to which the text  
        /// message will be sent.</param>  
        /// <param name="text">The text of the message to send.</param>
```

```
/// <returns></returns>
public async Task SendTextMessageAsync(string phoneNum, string text)
{
    if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
    {
        return;
    }

    // Now actually send the message.
    var request = new PublishRequest
    {
        Message = text,
        PhoneNumber = phoneNum
    };

    try
    {
        var response = await snsClient.PublishAsync(request);
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error sending message: {ex}");
    }
}

}
```

- For API details, see [Publish](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
 * Publish SMS: use Amazon SNS to send an SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 * NOTE: When you start using Amazon SNS to send SMS messages, your AWS account is
 * in the SMS sandbox and you can only
 * use verified destination phone numbers. See https://docs.aws.amazon.com/sns/latest/dg/sns-sms-sandbox.html.
 * NOTE: If destination is in the US, you also have an additional restriction that
 * you have use a dedicated
 * origination ID (phone number). You can request an origination number using
 * Amazon Pinpoint for a fee.
 * See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-origination-numbers-with-amazon-sns/
 * for more information.
 *
 * <phone_number_value> input parameter uses E.164 format.
 * For example, in United States, this input value should be of the form:
 +12223334444
 */
int main(int argc, char ** argv)
{
    if (argc != 3)
    {
```

```
    std::cout << "Usage: publish_sms <message_value> <phone_number_value> " <<
    std::endl;
    return 1;
}

Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;
    Aws::String message = argv[1];
    Aws::String phone_number = argv[2];

    Aws::SNS::Model::PublishRequest psms_req;
    psms_req.SetMessage(message);
    psms_req.SetPhoneNumber(phone_number);

    auto psms_out = sns.Publish(psms_req);

    if (psms_out.IsSuccess())
    {
        std::cout << "Message published successfully " <<
        psms_out.GetResult().GetMessageId()
        << std::endl;
    }
    else
    {
        std::cout << "Error while publishing message " <<
        psms_out.GetError().GetMessage()
        << std::endl;
    }
}

Aws::ShutdownAPI(options);
return 0;
}
```

- For API details, see [Publish](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .phoneNumber(phoneNumber)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out.println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- For API details, see [Publish](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun pubTextSMS(messageVal: String?, phoneNumberVal: String?) {  
  
    val request = PublishRequest {  
        message = messageVal  
        phoneNumber = phoneNumberVal  
    }  
  
    SnsClient { region = "us-east-1" }.use { snsClient -->  
        val result = snsClient.publish(request)  
        println("${result.messageId} message sent.")  
    }  
}
```

- For API details, see [Publish](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';  
  
use Aws\Sns\SnsClient;  
use Aws\Exception\AwsException;  
  
/**  
 * Sends a a text message (SMS message) directly to a phone number using Amazon  
 SNS.  
 *  
 * This code expects that you have AWS credentials set up per:  
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/  
 guide_credentials.html  
 */  
  
$SnsClient = new SnsClient([  
    'profile' => 'default',  
    'region' => 'us-east-1',  
    'version' => '2010-03-31'
```

```
]);
$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnsClient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [Publish](#) in [AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.

        :param phone_number: The phone number that receives the message. This must
        be
                           in E.164 format. For example, a United States phone
                           number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message)
            message_id = response['MessageId']
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id
```

- For API details, see [Publish](#) in [AWS SDK for Python \(Boto3\) API Reference](#).

## Publish to an Amazon SNS topic using an AWS SDK

The following code examples show how to publish messages to an Amazon SNS topic.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

- For API details, see [Publish](#) in *AWS SDK for .NET API Reference*.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;
    Aws::String message = argv[1];
    Aws::String topic_arn = argv[2];

    Aws::SNS::Model::PublishRequest psms_req;
    psms_req.SetMessage(message);
    psms_req.SetTopicArn(topic_arn);

    auto psms_out = sns.Publish(psms_req);

    if (psms_out.IsSuccess())
    {
        std::cout << "Message published successfully " << std::endl;
    }
    else
    {
        std::cout << "Error while publishing message " <<
        psms_out.GetError().GetMessage()
        << std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [Publish](#) in *AWS SDK for C++ API Reference*.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

- For API details, see [Publish](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void pubTopic(SnsClient snsClient, String message, String topicArn) {  
  
    try {  
        PublishRequest request = PublishRequest.builder()  
            .message(message)  
            .topicArn(topicArn)  
            .build();  
  
        PublishResponse result = snsClient.publish(request);  
        System.out.println(result.messageId() + " Message sent. Status is " +  
            result.sdkHttpResponse().statusCode());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [Publish](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create SNS service object.  
const snsClient = new SNSClient({ region: REGION });  
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js  
import {PublishCommand} from "@aws-sdk/client-sns";  
import {snsClient} from "./libs/snsClient.js";  
  
// Set the parameters  
var params = {  
    Message: "MESSAGE_TEXT", // MESSAGE_TEXT  
    TopicArn: "TOPIC_ARN", //TOPIC_ARN  
};
```

```
const run = async () => {
  try {
    const data = await snsClient.send(new PublishCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Publish](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun pubTopic(topicArnVal: String, messageVal: String) {

    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- For API details, see [Publish](#) in [AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Sends a message to an Amazon SNS topic.
 *
```

```
* This code expects that you have AWS credentials set up per:  
* https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/  
guide_credentials.html  
*/  
  
$SnSclient = new SnsClient([  
    'profile' => 'default',  
    'region' => 'us-east-1',  
    'version' => '2010-03-31'  
]);  
  
$message = 'This message is sent from a Amazon SNS code sample.';  
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';  
  
try {  
    $result = $SnSclient->publish([  
        'Message' => $message,  
        'TopicArn' => $topic,  
    ]);  
    var_dump($result);  
} catch (AwsException $e) {  
    // output error message if fails  
    error_log($e->getMessage());  
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [Publish](#) in [AWS SDK for PHP API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Publish a message with attributes so that a subscription can filter based on attributes.

```
class SnsWrapper:  
    """Encapsulates Amazon SNS topic and subscription functions."""  
    def __init__(self, sns_resource):  
        """  
        :param sns_resource: A Boto3 Amazon SNS resource.  
        """  
        self.sns_resource = sns_resource  
  
    def publish_message(topic, message, attributes):  
        """  
        Publishes a message, with attributes, to a topic. Subscriptions can be  
        filtered  
        based on message attributes so that a subscription receives messages only  
        when specified attributes are present.  
  
        :param topic: The topic to publish to.  
        :param message: The message to publish.  
        :param attributes: The key-value attributes to attach to the message.  
        Values  
            must be either `str` or `bytes`.  
        :return: The ID of the message.  
        """  
        try:  
            att_dict = {}
```

```

        for key, value in attributes.items():
            if isinstance(value, str):
                att_dict[key] = {'DataType': 'String', 'StringValue': value}
            elif isinstance(value, bytes):
                att_dict[key] = {'DataType': 'Binary', 'BinaryValue': value}
        response = topic.publish(Message=message, MessageAttributes=att_dict)
        message_id = response['MessageId']
        logger.info(
            "Published message with attributes %s to topic %s.", attributes,
            topic.arn)
    except ClientError:
        logger.exception("Couldn't publish message to topic %s.", topic.arn)
        raise
    else:
        return message_id
    
```

Publish a message that takes different forms based on the protocol of the subscriber.

```

class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def publish_multi_message(
            topic, subject, default_message, sms_message, email_message):
        """
        Publishes a multi-format message to a topic. A multi-format message takes
        different forms based on the protocol of the subscriber. For example,
        an SMS subscriber might receive a short, text-only version of the message
        while an email subscriber could receive an HTML version of the message.

        :param topic: The topic to publish to.
        :param subject: The subject of the message.
        :param default_message: The default version of the message. This version is
                               sent to subscribers that have protocols that are
                               not
                               otherwise specified in the structured message.
        :param sms_message: The version of the message sent to SMS subscribers.
        :param email_message: The version of the message sent to email subscribers.
        :return: The ID of the message.
        """
        try:
            message = {
                'default': default_message,
                'sms': sms_message,
                'email': email_message
            }
            response = topic.publish(
                Message=json.dumps(message), Subject=subject,
                MessageStructure='json')
            message_id = response['MessageId']
            logger.info("Published multi-format message to topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
    
```

- For API details, see [Publish in AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-sns" # v2: require 'aws-sdk'

def message_sent?(sns_client, topic_arn, message)

    sns_client.publish(topic_arn: topic_arn, message: message)
rescue StandardError => e
    puts "Error while sending the message: #{e.message}"
end

def run_me

    topic_arn = "SNS_TOPIC_ARN"
    region = "REGION"
    message = "MESSAGE" # The text of the message to send.

    sns_client = Aws::SNS::Client.new(region: region)

    puts "Message sending."

    if message_sent?(sns_client, topic_arn, message)
        puts "The message was sent."
    else
        puts "The message was not sent. Stopping program."
        exit 1
    end
end

run_me if $PROGRAM_NAME == __FILE__
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [Publish in AWS SDK for Ruby API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
```

```
    println!("Receiving on topic with ARN: `{}`, topic_arn);  
  
    let rsp = client  
        .subscribe()  
        .topic_arn(topic_arn)  
        .protocol("email")  
        .endpoint(email_address)  
        .send()  
        .await?;  
  
    println!("Added a subscription: {:?}", rsp);  
  
    let rsp = client  
        .publish()  
        .topic_arn(topic_arn)  
        .message("hello sns!")  
        .send()  
        .await?;  
  
    println!("Published message: {:?}", rsp);  
  
    Ok(())
}
```

- For API details, see [Publish](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sns->publish(                                     " oo_result is returned for  
testing purpose "  
        iv_topicarn = iv_topic_arn  
        iv_message = iv_message  
    ).  
    MESSAGE 'Message published to SNS topic' TYPE 'I'.  
CATCH /aws1/cx_snsnotfoundexception.  
    MESSAGE 'Topic does not exist' TYPE 'E'.  
ENDTRY.
```

- For API details, see [Publish](#) in *AWS SDK for SAP ABAP API reference*.

## Set a dead-letter queue for an Amazon SNS subscription using an AWS SDK

The following code example shows how to set an Amazon SQS queue as a dead-letter queue for an Amazon SNS subscription.

Java

### SDK for Java 1.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Specify the ARN of the Amazon SNS subscription.  
String subscriptionArn =  
    "arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-  
bc89-012d-3e45-6fg7h890123i";  
  
// Specify the ARN of the Amazon SQS queue to use as a dead-letter queue.  
String redrivePolicy =  
    "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-  
east-2:123456789012:MyDeadLetterQueue\"}";  
  
// Set the specified Amazon SQS queue as a dead-letter queue  
// of the specified Amazon SNS subscription by setting the RedrivePolicy attribute.  
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()  
    .withSubscriptionArn(subscriptionArn)  
    .withAttributeName("RedrivePolicy")  
    .withAttributeValue(redrivePolicy);  
sns.setSubscriptionAttributes(request);
```

## Set an Amazon SNS filter policy using an AWS SDK

The following code examples show how to set an Amazon SNS filter policy.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void usePolicy(SnsClient snsClient, String subscriptionArn) {  
  
    try {  
        SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();  
        // Add a filter policy attribute with a single value  
        fp.addAttribute("store", "example_corp");  
        fp.addAttribute("event", "order_placed");  
  
        // Add a prefix attribute  
        fp.addAttributePrefix("customer_interests", "bas");  
  
        // Add an anything-but attribute  
        fp.addAttributeAnythingBut("customer_interests", "baseball");  
  
        // Add a filter policy attribute with a list of values  
        ArrayList<String> attributeValues = new ArrayList<>();  
        attributeValues.add("rugby");  
        attributeValues.add("soccer");  
        attributeValues.add("hockey");  
        fp.addAttribute("customer_interests", attributeValues);  
    } catch (AmazonServiceException ase) {  
        System.out.println("Caught an AmazonServiceException, which means your request failed to reach the AWS Service." +  
            "\nError Message: " + ase.getMessage());  
        System.out.println("HTTP Status Code: " + ase.getStatusCode());  
        System.out.println("AWS Error Code: " + ase.getErrorCode());  
        System.out.println("Request ID: " + ase.getRequestId());  
    } catch (AmazonClientException ace) {  
        System.out.println("Caught an AmazonClientException, which means the client encountered a serious internal problem." +  
            "\nError Message: " + ace.getMessage());  
    }  
}
```

```
// Add a numeric attribute
fp.addAttribute("price_usd", "=", 0);

// Add a numeric attribute with a range
fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

// Apply the filter policy attributes to an Amazon SNS subscription
fp.apply(snsClient, subscriptionArn);

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [SetSubscriptionAttributes](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def add_subscription_filter(subscription, attributes):
        """
        Adds a filter policy to a subscription. A filter policy is a key and a
        list of values that are allowed. When a message is published, it must have
        an
        attribute that passes the filter or it will not be sent to the
        subscription.

        :param subscription: The subscription the filter policy is attached to.
        :param attributes: A dictionary of key-value pairs that define the filter.
        """
        try:
            att_policy = {key: [value] for key, value in attributes.items()}
            subscription.set_attributes(
                AttributeName='FilterPolicy',
                AttributeValue=json.dumps(att_policy))
            logger.info("Added filter to subscription %s.", subscription.arn)
        except ClientError:
            logger.exception(
                "Couldn't add filter to subscription %s.", subscription.arn)
            raise
```

- For API details, see [SetSubscriptionAttributes](#) in *AWS SDK for Python (Boto3) API Reference*.

## Set the default settings for sending Amazon SNS SMS messages using an AWS SDK

The following code examples show how to set the default settings for sending SMS messages using Amazon SNS.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

How to use Amazon SNS to set the DefaultSMSType attribute.

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;
    Aws::String sms_type = argv[1];

    Aws::SNS::Model::SetSMSAttributesRequest ssmst_req;
    ssmst_req.AddAttributes("DefaultSMSType", sms_type);

    auto ssmst_out = sns.SetSMSAttributes(ssmst_req);

    if (ssmst_out.IsSuccess())
    {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else
    {
        std::cout << "Error while setting SMS Type: '" <<
        ssmst_out.GetError().GetMessage()
        << "'" << std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [SetSmsAttributes](#) in *AWS SDK for C++ API Reference*.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public class SetSMSAttributes {
    public static void main(String[] args) {

        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket" );
```

```
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();
setSNSAttributes(snsClient, attributes);
snsClient.close();
}

public static void setSNSAttributes( SnsClient snsClient, HashMap<String,
String> attributes) {

    try {
        SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
            .attributes(attributes)
            .build();

        SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
        System.out.println("Set default Attributes to " + attributes + ".
Status was " + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SetSmsAttributes](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {SetSMSAttributesCommand } from "@aws-sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const params = {
    attributes: {
        /* required */
        DefaultSMSType: "Transactional" /* highest reliability */,
        //DefaultSMSType: 'Promotional' /* lowest cost */
    },
};
```

```
const run = async () => {
  try {
    const data = await snsClient.send(new SetSMSAttributesCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SetSmsAttributes](#) in [AWS SDK for JavaScript API Reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$SnSclient = new SnsClient([
  'profile' => 'default',
  'region' => 'us-east-1',
  'version' => '2010-03-31'
]);

try {
  $result = $SnSclient->SetSMSAttributes([
    'attributes' => [
      'DefaultSMSType' => 'Transactional',
    ],
  ]);
  var_dump($result);
} catch (AwsException $e) {
  // output error message if fails
  error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [SetSmsAttributes](#) in [AWS SDK for PHP API Reference](#).

## Set Amazon SNS topic attributes using an AWS SDK

The following code examples show how to set Amazon SNS topic attributes.

### Java

#### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void setTopAttr(SnsClient snsClient, String attribute, String topicArn, String value) {

        try {
            SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
                .attributeName(attribute)
                .attributeValue(value)
                .topicArn(topicArn)
                .build();

            SetTopicAttributesResponse result =
snsClient.setTopicAttributes(request);
System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode() + "\n\nTopic " + request.topicArn()
+ " updated " + request.attributeName() + " to " +
request.attributeValue());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [SetTopicAttributes](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {SetTopicAttributesCommand } from "@aws-sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const params = {
    AttributeName: "ATTRIBUTE_NAME", // ATTRIBUTE_NAME
    TopicArn: "TOPIC_ARN", // TOPIC_ARN
    AttributeValue: "NEW_ATTRIBUTE_VALUE", //NEW_ATTRIBUTE_VALUE
};

const run = async () => {
    try {
```

```
    const data = await snsClient.send(new SetTopicAttributesCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
} catch (err) {
    console.log("Error", err.stack);
}
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SetTopicAttributes](#) in *AWS SDK for JavaScript API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun setTopAttr(attribute: String?, topicArnVal: String?, value: String?) {

    val request = SetTopicAttributesRequest {
        attributeName = attribute
        attributeValue = value
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- For API details, see [SetTopicAttributes](#) in *AWS SDK for Kotlin API reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
```

```
* This code expects that you have AWS credentials set up per:  
* https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/  
guide_credentials.html  
*/  
  
$SnSclient = new SnsClient([  
    'profile' => 'default',  
    'region' => 'us-east-1',  
    'version' => '2010-03-31'  
]);  
$attribute = 'Policy | DisplayName | DeliveryPolicy';  
$value = 'First Topic';  
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';  
  
try {  
    $result = $SnSclient->setTopicAttributes([  
        'AttributeName' => $attribute,  
        'AttributeValue' => $value,  
        'TopicArn' => $topic,  
    ]);  
    var_dump($result);  
} catch (AwsException $e) {  
    // output error message if fails  
    error_log($e->getMessage());  
}
```

- For API details, see [SetTopicAttributes](#) in *AWS SDK for PHP API Reference*.

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-sns" # v2: require 'aws-sdk'  
  
policy = {  
    "Version": "2008-10-17",  
    "Id": "__default_policy_ID",  
    "Statement": [  
        {  
            "Sid": "__default_statement_ID",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "*"  
            },  
            "Action": ["SNS:Publish"],  
            "Resource": "' + MY_TOPIC_ARN + ''",  
            "Condition": {  
                "ArnEquals": {  
                    "AWS:SourceArn": "' + MY_RESOURCE_ARN + ''"  
                }  
            }  
        }  
    ]  
}  
# Replace us-west-2 with the AWS Region you're using for Amazon SNS.  
sns = Aws::SNS::Resource.new(region: "REGION")  
  
# Get topic by ARN  
topic = sns.topic()
```

```
# Add policy to topic
topic.set_attributes({
    attribute_name: "POLICY_NAME",
    attribute_value: policy
})
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [SetTopicAttributes](#) in [AWS SDK for Ruby API Reference](#).

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    lo_sns->settopicattributes(
        iv_topicarn = iv_topic_arn
        iv_attributename = iv_attribute_name
        iv_attributevalue = iv_attribute_value
    ).
    MESSAGE 'Set/Updated SNS topic attributes' TYPE 'I'.
CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist' TYPE 'E'.
ENDTRY.
```

- For API details, see [SetTopicAttributes](#) in [AWS SDK for SAP ABAP API reference](#).

## Subscribe a Lambda function to receive notifications from an Amazon SNS topic using an AWS SDK

The following code examples show how to subscribe a Lambda function so it receives notifications from an Amazon SNS topic.

### C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
 * Subscribe an AWS Lambda endpoint to a topic - demonstrates how to initiate a
 * subscription to an Amazon SNS topic with delivery
 * to an AWS Lambda function.
```

```
*  
* NOTE: You must first configure AWS Lambda to run this example.  
* See https://docs.amazonaws.cn/en\_us/lambda/latest/dg/with-sns-example.html for more information.  
*  
* <protocol_value> set to "lambda" provides delivery of JSON-encoded message to an  
AWS Lambda function  
* (see https://docs.aws.amazon.com/sns/latest/api/API\_Subscribe.html for  
available protocols).  
* <topic_arn_value> can be obtained from run_list_topics executable and includes  
the "arn:" prefix.  
* <lambda_function_arn> is the ARN of an AWS Lambda function.  
*/  
  
int main(int argc, char ** argv)  
{  
    if (argc != 4)  
    {  
        std::cout << "Usage: subscribe_lambda <protocol_value=lambda> <topic_arn_value>"  
              " <lambda_function_arn>" << std::endl;  
        return 1;  
    }  
  
    Aws::SDKOptions options;  
    Aws::InitAPI(options);  
    {  
        Aws::SNS::SNSClient sns;  
        Aws::String protocol = argv[1];  
        Aws::String topic_arn = argv[2];  
        Aws::String endpoint = argv[3];  
  
        Aws::SNS::Model::SubscribeRequest s_req;  
        s_req.SetTopicArn(topic_arn);  
        s_req.SetProtocol(protocol);  
        s_req.SetEndpoint(endpoint);  
  
        auto s_out = sns.Subscribe(s_req);  
  
        if (s_out.IsSuccess())  
        {  
            std::cout << "Subscribed successfully " << std::endl;  
        }  
        else  
        {  
            std::cout << "Error while subscribing " << s_out.GetError().GetMessage()  
                  << std::endl;  
        }  
    }  
  
    Aws::ShutdownAPI(options);  
    return 0;  
}
```

- For API details, see [Subscribe](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static String subLambda(SnsClient snsClient, String topicArn, String lambdaArn) {  
  
        try {  
            SubscribeRequest request = SubscribeRequest.builder()  
                .protocol("lambda")  
                .endpoint(lambdaArn)  
                .returnSubscriptionArn(true)  
                .topicArn(topicArn)  
                .build();  
  
            SubscribeResponse result = snsClient.subscribe(request);  
            return result.subscriptionArn();  
  
        } catch (SnsException e) {  
            System.err.println(e.awsErrorDetails().errorMessage());  
            System.exit(1);  
        }  
        return "";  
    }  
}
```

- For API details, see [Subscribe](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create SNS service object.  
const snsClient = new SNSClient({ region: REGION });  
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js  
import {SubscribeCommand} from "@aws-sdk/client-sns";  
import {snsClient} from "./libs/snsClient.js";  
  
// Set the parameters  
const params = {  
    Protocol: "lambda" /* required */,  
    TopicArn: "TOPIC_ARN", //TOPIC_ARN  
    Endpoint: "LAMBDA_FUNCTION_ARN", //LAMBDA_FUNCTION_ARN  
};  
  
const run = async () => {  
    try {  
        const data = await snsClient.send(new SubscribeCommand(params));  
        console.log("Success.", data);  
        return data; // For unit tests.  
    } catch (err) {
```

```
        console.log("Error", err.stack);
    }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Subscribe](#) in [AWS SDK for JavaScript API Reference](#).

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun subLambda(topicArnVal: String?, lambdaArn: String?) {

    val request = SubscribeRequest {
        protocol = "lambda"
        endpoint = lambdaArn
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- For API details, see [Subscribe](#) in [AWS SDK for Kotlin API reference](#).

## Subscribe a mobile application to an Amazon SNS topic using an AWS SDK

The following code examples show how to subscribe a mobile application endpoint so it receives notifications from an Amazon SNS topic.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
 * Subscribe an app endpoint to a topic - demonstrates how to initiate a
 * subscription to an Amazon SNS topic
```

```
* with delivery to a mobile app.
*
* NOTE: You must first create an endpoint by registering an app and device.
* See https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-
devicetoken.html for more information.
*
* <protocol_value> set to "application" provides delivery of JSON-encoded message
to an EndpointArn
* for a mobile app and device (see https://docs.aws.amazon.com/sns/latest/
api/API_Subscribe.html for available protocols).
* <topic_arn_value> can be obtained from run_list_topics executable and includes
the "arn:" prefix.
* <mobile_endpoint_arn> is the EndpointArn of a mobile app and device.
*/
int main(int argc, char ** argv)
{
    if (argc != 4)
    {
        std::cout << "Usage: subscribe_app <protocol_value/application>
<topic_arn_value>"<< " <mobile_endpoint_arn>" << std::endl;
        return 1;
    }

    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        Aws::SNS::SNSClient sns;
        Aws::String protocol = argv[1];
        Aws::String topic_arn = argv[2];
        Aws::String endpoint = argv[3];

        Aws::SNS::Model::SubscribeRequest s_req;
        s_req.SetTopicArn(topic_arn);
        s_req.SetProtocol(protocol);
        s_req.SetEndpoint(endpoint);

        auto s_out = sns.Subscribe(s_req);

        if (s_out.IsSuccess())
        {
            std::cout << "Subscribed successfully " << std::endl;
        }
        else
        {
            std::cout << "Error while subscribing " << s_out.GetError().GetMessage()
                << std::endl;
        }
    }

    Aws::ShutdownAPI(options);
    return 0;
}
```

- For API details, see [Subscribe](#) in *AWS SDK for C++ API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {SubscribeCommand} from "@aws-sdk/client-sns";
import {snsClient} from "./libs/snsClient.js";

// Set the parameters
const params = {
  Protocol: "application" /* required */,
  TopicArn: "TOPIC_ARN", //TOPIC_ARN
  Endpoint: "MOBILE_ENDPOINT_ARN", // MOBILE_ENDPOINT_ARN
};

const run = async () => {
  try {
    const data = await snsClient.send(new SubscribeCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Subscribe](#) in [AWS SDK for JavaScript API Reference](#).

## Subscribe an HTTP endpoint to an Amazon SNS topic using an AWS SDK

The following code examples show how to subscribe an HTTP or HTTPS endpoint so it receives notifications from an Amazon SNS topic.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void subHTTPS(SnsClient snsClient, String topicArn, String url )
{
  try {
    SubscribeRequest request = SubscribeRequest.builder()
```

```
.protocol("https")
.endpoint(url)
.returnSubscriptionArn(true)
.topicArn(topicArn)
.build();

SubscribeResponse result = snsClient.subscribe(request);
System.out.println("Subscription ARN is " + result.subscriptionArn() +
"\n\n Status is " + result.sdkHttpResponse().statusCode());

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [Subscribe](#) in *AWS SDK for Java 2.x API Reference*.

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 * message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
```

```
}
```

- For API details, see [Subscribe](#) in *AWS SDK for PHP API Reference*.

## Subscribe an email address to an Amazon SNS topic using an AWS SDK

The following code examples show how to subscribe an email address to an Amazon SNS topic.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Creates a new subscription to a topic.
///</summary>
///<param name="client">The initialized Amazon SNS client object, used
///to create an Amazon SNS subscription.</param>
///<param name="topicArn">The ARN of the topic to subscribe to.</param>
///<returns>A SubscribeResponse object which includes the subscription
///ARN for the new subscription.</returns>
public static async Task<SubscribeResponse> TopicSubscribeAsync(
    IAmazonSimpleNotificationService client,
    string topicArn)
{
    SubscribeRequest request = new SubscribeRequest()
    {
        TopicArn = topicArn,
        ReturnSubscriptionArn = true,
        Protocol = "email",
        Endpoint = "recipient@example.com",
    };

    var response = await client.SubscribeAsync(request);

    return response;
}
```

- For API details, see [Subscribe](#) in *AWS SDK for .NET API Reference*.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
```

```
* Subscribe an email address endpoint to a topic - demonstrates how to initiate a
subscription to an Amazon SNS topic with delivery
*   to an email address.
*
* SNS will send a subscription confirmation email to the email address provided
which you need to confirm to
* receive messages.
*
* <protocol_value> set to "email" provides delivery of message via SMTP (see
https://docs.aws.amazon.com/sns/latest/api/API_Subscribe.html for available
protocols).
* <topic_arn_value> can be obtained from run_list_topics executable and includes
the "arn:" prefix.
*/
int main(int argc, char ** argv)
{
    if (argc != 4)
    {
        std::cout << "Usage: subscribe_email <protocol_value=email> <topic_arn_value>"<< std::endl;
        return 1;
    }

    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        Aws::SNS::SNSClient sns;
        Aws::String protocol = argv[1];
        Aws::String topic_arn = argv[2];
        Aws::String endpoint = argv[3];

        Aws::SNS::Model::SubscribeRequest s_req;
        s_req.SetTopicArn(topic_arn);
        s_req.SetProtocol(protocol);
        s_req.SetEndpoint(endpoint);

        auto s_out = sns.Subscribe(s_req);

        if (s_out.IsSuccess())
        {
            std::cout << "Subscribed successfully " << std::endl;
        }
        else
        {
            std::cout << "Error while subscribing " << s_out.GetError().GetMessage()
                << std::endl;
        }
    }

    Aws::ShutdownAPI(options);
    return 0;
}
```

- For API details, see [Subscribe](#) in *AWS SDK for C++ API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

- For API details, see [Subscribe](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void subEmail(SnsClient snsClient, String topicArn, String email)
{
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("email")
            .endpoint(email)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN: " + result.subscriptionArn() +
            "\n\n Status is " + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [Subscribe](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {SubscribeCommand } from "@aws-sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
```

```
const params = {
  Protocol: "email" /* required */,
  TopicArn: "TOPIC_ARN", //TOPIC_ARN
  Endpoint: "EMAIL_ADDRESS", //EMAIL_ADDRESS
};

const run = async () => {
  try {
    const data = await snsClient.send(new SubscribeCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Subscribe](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun subEmail(topicArnVal: String, email: String): String {

    val request = SubscribeRequest {
        protocol = "email"
        endpoint = email
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient -->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

- For API details, see [Subscribe](#) in [AWS SDK for Kotlin API reference](#).

## PHP

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';
```

```
use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 * message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For API details, see [Subscribe](#) in *AWS SDK for PHP API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def subscribe(topic, protocol, endpoint):
        """
        Subscribes an endpoint to the topic. Some endpoint types, such as email,
        must be confirmed before their subscriptions are active. When a
        subscription
            is not confirmed, its Amazon Resource Number (ARN) is set to
            'PendingConfirmation'.
        """

```

```
:param topic: The topic to subscribe to.
:param protocol: The protocol of the endpoint, such as 'sms' or 'email'.
:param endpoint: The endpoint that receives messages, such as a phone
number
                           (in E.164 format) for SMS messages, or an email address
for
                           email messages.
:return: The newly added subscription.
"""
try:
    subscription = topic.subscribe(
        Protocol=protocol, Endpoint=endpoint, ReturnSubscriptionArn=True)
    logger.info("Subscribed %s %s to topic %s.", protocol, endpoint,
topic.arn)
except ClientError:
    logger.exception(
        "Couldn't subscribe %s %s to topic %s.", protocol, endpoint,
topic.arn)
    raise
else:
    return subscription
```

- For API details, see [Subscribe in AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-sns" # v2: require 'aws-sdk'

def subscription_created?(sns_client, topic_arn, protocol, endpoint)

    sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
endpoint)

rescue StandardError => e
    puts "Error while creating the subscription: #{e.message}"
end

# Full example call:
def run_me

    protocol = "email"
endpoint = "EMAIL_ADDRESS"
topic_arn = "TOPIC_ARN"
region = "REGION"

sns_client = Aws::SNS::Client.new(region: region)

puts "Creating the subscription."

if subscription_created?(sns_client, topic_arn, protocol, endpoint)
    puts "The subscription was created."
else
    puts "The subscription was not created. Stopping program."
    exit 1
end
```

```
end

run_me if $PROGRAM_NAME == __FILE__
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [Subscribe](#) in [AWS SDK for Ruby API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}'", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);
}

Ok(())
}
```

- For API details, see [Subscribe](#) in [AWS SDK for Rust API reference](#).

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sns->subscribe()  
    "oo_result is returned  
for testing purpose"  
    iv_topicarn = iv_topic_arn  
    iv_protocol = 'email'  
    iv_endpoint = iv_email_address  
    iv_returnsubscriptionarn = abap_true  
).  
MESSAGE 'Email address subscribed to SNS topic' TYPE 'I'.  
CATCH /aws1/cx_snsnotfoundexception.  
    MESSAGE 'Topic does not exist' TYPE 'E'.  
CATCH /aws1/cx_snssubscriptionlmtex00.  
    MESSAGE 'Unable to create subscriptions, you have reached the maximum  
number of subscriptions allowed.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [Subscribe](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios for Amazon SNS using AWS SDKs

The following code examples show how to use Amazon Simple Notification Service with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Create a platform endpoint for Amazon SNS push notifications using an AWS SDK \(p. 2007\)](#)
- [Create and publish to a FIFO Amazon SNS topic using an AWS SDK \(p. 2008\)](#)
- [Publish SMS messages to an Amazon SNS topic using an AWS SDK \(p. 2011\)](#)
- [Publish a large message to Amazon SNS with Amazon S3 using an AWS SDK \(p. 2013\)](#)

## Create a platform endpoint for Amazon SNS push notifications using an AWS SDK

The following code example shows how to create a platform endpoint for Amazon SNS push notifications.

Java

### SDK for Java 2.x

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public class RegistrationExample {  
  
    public static void main(String[] args) {  
  
        final String usage = "\n" +  
            "Usage: " +
```

```
"      <token>\n\n" +
"Where:\n" +
"  token - The name of the FIFO topic. \n\n" +
"  platformApplicationArn - The ARN value of platform application. You
can get this value from the AWS Management Console. \n\n";
if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String token = args[0];
String platformApplicationArn = args[1];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

createEndpoint(snsClient, token, platformApplicationArn);
}

public static void createEndpoint(SnsClient snsClient, String token, String
platformApplicationArn){

    System.out.println("Creating platform endpoint with token " + token);

    try {
        CreatePlatformEndpointRequest endpointRequest =
CreatePlatformEndpointRequest.builder()
        .token(token)
        .platformApplicationArn(platformApplicationArn)
        .build();

        CreatePlatformEndpointResponse response =
snsClient.createPlatformEndpoint(endpointRequest);
        System.out.println("The ARN of the endpoint is " +
response.endpointArn());
    } catch ( SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

## Create and publish to a FIFO Amazon SNS topic using an AWS SDK

The following code examples show how to create and publish to a FIFO Amazon SNS topic.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a FIFO topic and FIFO queues. Subscribe the queues to the topic.

```
public static void main(String[] args) {
```

```
final String usage = "\n" +
    "Usage: " +
    "    <topicArn>\n\n" +
    "Where:\n" +
    "    fifoTopicName - The name of the FIFO topic. \n\n" +
    "    fifoQueueARN - The ARN value of a SQS FIFO queue. You can get this
value from the AWS Management Console. \n\n";

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String fifoTopicName = "PriceUpdatesTopic3 fifo";
String fifoQueueARN = "arn:aws:sqs:us-east-1:814548047983:MyPriceSQS fifo";
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

createFIFO(snsClient, fifoTopicName, fifoQueueARN);
}

public static void createFIFO(SnsClient snsClient, String topicName, String
queueARN) {

    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = new HashMap<>();
        topicAttributes.put("FifoTopic", "true");
        topicAttributes.put("ContentBasedDeduplication", "false");

        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
            .name(topicName)
            .attributes(topicAttributes)
            .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        String topicArn = response.topicArn();
        System.out.println("The topic ARN is"+topicArn);

        // Subscribe to the endpoint by using the SNS service client.
        // Only Amazon SQS FIFO queues can receive notifications from an Amazon
SNS FIFO topic.
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicArn)
            .endpoint(queueARN)
            .protocol("sqS")
            .build();

        snsClient.subscribe(subscribeRequest);
        System.out.println("The topic is subscribed to the queue.");

        // Compose and publish a message that updates the wholesale price.
        String subject = "Price Update";
        String payload = "{\"product\": 214, \"price\": 79.99}";
        String groupId = "PID-214";
        String dedupId = UUID.randomUUID().toString();
        String attributeName = "business";
        String attributeValue = "wholesale";

        MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
            .dataType("String")
            .stringValue(attributeValue)
            .build();
    }
}
```

```
Map<String, MessageAttributeValue> attributes = new HashMap<>();
attributes.put(attributeName, msgAttValue);
PublishRequest pubRequest = PublishRequest.builder()
    .topicArn(topicArn)
    .subject(subject)
    .message(payload)
    .messageGroupId(groupId)
    .messageDuplicationId(dedupId)
    .messageAttributes(attributes)
    .build();

snsClient.publish(pubRequest);
System.out.println("Message was published to "+topicArn);

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a FIFO topic, subscribe an Amazon SQS FIFO queue to the topic, and publish a message to an Amazon SNS topic.

```
" Creates a FIFO topic "
DATA lt_tpc_attributes TYPE /aws1/cl_snstopicattrsm_w=>tt_topicattributesmap.
DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsm_w=>ts_topicattributesmap_maprow.
ls_tpc_attributes-key = 'FifoTopic'.
ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsm_w( iv_value = 'true' ).
INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

TRY.
    DATA(lo_create_result) = lo_sns->createtopic(
        iv_name = iv_topic_name
        it_attributes = lt_tpc_attributes
    ).
    DATA(lv_topic_arn) = lo_create_result->get_topicarn( )."
    ov_topic_arn = lv_topic_arn.
    ov_topic_arn is returned for testing purpose "
    MESSAGE 'FIFO topic created' TYPE 'I'.
    CATCH /aws1/cx_snstopiclimitcxdex.
        MESSAGE 'Unable to create more topics as you have reached the maximum
number of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon SNS topic "
    " Only SQS FIFO queues can be subscribed to an SNS FIFO topic "
TRY.
    DATA(lo_subscribe_result) = lo_sns->subscribe(
```

```
        iv_topicarn = lv_topic_arn
        iv_protocol = 'sqS'
        iv_endpoint = iv_queue_arn
    ).
    DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
    ov_subscription_arn = lv_subscription_arn.
    ov_subscription_arn is returned for testing purpose "
        MESSAGE 'SQS Queue was subscribed to SNS topic' TYPE 'I'.
    CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist' TYPE 'E'.
    CATCH /aws1/cx_snssubscriptionlmt00.
        MESSAGE 'Unable to create subscriptions, you have reached the maximum
number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
'String' iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
            iv_topicarn = lv_topic_arn
            iv_message = 'The price of your mobile plan has been increased from
$19 to $23'
            iv_subject = 'Changes to mobile plan'
            iv_messagegroupid = 'Update-2'
            iv_messagededuplicationid = 'Update-2.1'
            it_messageattributes = lt_msg_attributes
        ).
        ov_message_id = lo_result->get_messageid( ).
        ov_message_id is returned for testing purpose "
            MESSAGE 'Message was published to SNS topic' TYPE 'I'.
        CATCH /aws1/cx_snsnotfoundexception.
            MESSAGE 'Topic does not exist' TYPE 'E'.
    ENDTRY.
```

## Publish SMS messages to an Amazon SNS topic using an AWS SDK

The following code example shows how to:

- Create an Amazon SNS topic.
- Subscribe phone numbers to the topic.
- Publish SMS messages to the topic so that all subscribed phone numbers receive the message at once.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a topic and return its ARN.

```
public static String createSNSTopic(SnsClient snsClient, String topicName ) {  
  
    CreateTopicResponse result = null;  
    try {  
        CreateTopicRequest request = CreateTopicRequest.builder()  
            .name(topicName)  
            .build();  
  
        result = snsClient.createTopic(request);  
        return result.topicArn();  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

Subscribe an endpoint to a topic.

```
public static void subTextSNS( SnsClient snsClient, String topicArn, String  
phoneNumber) {  
  
    try {  
        SubscribeRequest request = SubscribeRequest.builder()  
            .protocol("sms")  
            .endpoint(phoneNumber)  
            .returnSubscriptionArn(true)  
            .topicArn(topicArn)  
            .build();  
  
        SubscribeResponse result = snsClient.subscribe(request);  
        System.out.println("Subscription ARN: " + result.subscriptionArn() +  
"\n\n Status is " + result.sdkHttpResponse().statusCode());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

Set attributes on the message, such as the ID of the sender, the maximum price, and its type.  
Message attributes are optional.

```
public class SetSMSAttributes {  
    public static void main(String[] args) {  
  
        HashMap<String, String> attributes = new HashMap<>(1);  
        attributes.put("DefaultSMSType", "Transactional");  
        attributes.put("UsageReportS3Bucket", "janbucket" );  
  
        SnsClient snsClient = SnsClient.builder()  
            .region(Region.US_EAST_1)  
            .credentialsProvider(ProfileCredentialsProvider.create())  
            .build();  
        setSNSAttributes(snsClient, attributes);  
        snsClient.close();  
    }  
}
```

```
public static void setSNSAttributes( SnsClient snsClient, HashMap<String, String> attributes) {  
  
    try {  
        SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()  
            .attributes(attributes)  
            .build();  
  
        SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);  
        System.out.println("Set default Attributes to " + attributes + ".  
Status was " + result.sdkHttpResponse().statusCode());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

Publish a message to a topic. The message is sent to every subscriber.

```
public static void pubTextSMS(SnsClient snsClient, String message, String phoneNumber) {  
    try {  
        PublishRequest request = PublishRequest.builder()  
            .message(message)  
            .phoneNumber(phoneNumber)  
            .build();  
  
        PublishResponse result = snsClient.publish(request);  
        System.out.println(result.messageId() + " Message sent. Status was " +  
result.sdkHttpResponse().statusCode());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

## Publish a large message to Amazon SNS with Amazon S3 using an AWS SDK

The following code example shows how to publish a large message to Amazon SNS using Amazon S3 to store the message payload.

Java

### SDK for Java 1.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

To publish a large message, use the Amazon SNS Extended Client Library for Java. The message that you send references an Amazon S3 object containing the actual message content.

```
import com.amazonaws.sqs.javamessaging.AmazonSQSExtendedClient;  
import com.amazonaws.sqs.javamessaging.ExtendedClientConfiguration;
```

```
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.PublishRequest;
import com.amazonaws.services.sns.model.SetSubscriptionAttributesRequest;
import com.amazonaws.services.sns.util.Topics;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
import software.amazon sns AmazonSNSExtendedClient;
import software.amazon sns SNSExtendedClientConfiguration;

public class Example {

    public static void main(String[] args) {
        final String BUCKET_NAME = "extended-client-bucket";
        final String TOPIC_NAME = "extended-client-topic";
        final String QUEUE_NAME = "extended-client-queue";
        final Regions region = Regions.DEFAULT_REGION;

        //Message threshold controls the maximum message size that will be allowed
        to be published
        //through SNS using the extended client. Payload of messages exceeding this
        value will be stored in
        //S3. The default value of this parameter is 256 KB which is the maximum
        message size in SNS (and SQS).
        final int EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD = 32;

        //Initialize SNS, SQS and S3 clients
        final AmazonSNS snsClient =
AmazonSNSClientBuilder.standard().withRegion(region).build();
        final AmazonSQS sqsClient =
AmazonSQSClientBuilder.standard().withRegion(region).build();
        final AmazonS3 s3Client =
AmazonS3ClientBuilder.standard().withRegion(region).build();

        //Create bucket, topic, queue and subscription
        s3Client.createBucket(BUCKET_NAME);
        final String topicArn = snsClient.createTopic(
            new CreateTopicRequest().withName(TOPIC_NAME)
        ).getTopicArn();
        final String queueUrl = sqsClient.createQueue(
            new CreateQueueRequest().withQueueName(QUEUE_NAME)
        ).getQueueUrl();
        final String subscriptionArn = Topics.subscribeQueue(
            snsClient, sqsClient, topicArn, queueUrl
        );

        //To read message content stored in S3 transparently through SQS extended
        client,
        //set the RawMessageDelivery subscription attribute to TRUE
        final SetSubscriptionAttributesRequest subscriptionAttributesRequest = new
SetSubscriptionAttributesRequest();
        subscriptionAttributesRequest.setSubscriptionArn(subscriptionArn);
        subscriptionAttributesRequest.setAttributeName("RawMessageDelivery");
        subscriptionAttributesRequest.setAttributeValue("TRUE");
        snsClient.setSubscriptionAttributes(subscriptionAttributesRequest);

        //Initialize SNS extended client
        //PayloadSizeThreshold triggers message content storage in S3 when the
        threshold is exceeded
    }
}
```

```
//To store all messages content in S3, use AlwaysThroughS3 flag
final SNSExtendedClientConfiguration snsExtendedClientConfiguration = new
SNSExtendedClientConfiguration()
    .withPayloadSupportEnabled(s3Client, BUCKET_NAME)
    .withPayloadSizeThreshold(EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD);
final AmazonSNSExtendedClient snsExtendedClient = new
AmazonSNSExtendedClient(snsClient, snsExtendedClientConfiguration);

//Publish message via SNS with storage in S3
final String message = "This message is stored in S3 as it exceeds the
threshold of 32 bytes set above.";
snsExtendedClient.publish(topicArn, message);

//Initialize SQS extended client
final ExtendedClientConfiguration sqsExtendedClientConfiguration = new
ExtendedClientConfiguration()
    .withPayloadSupportEnabled(s3Client, BUCKET_NAME);
final AmazonSQSExtendedClient sqsExtendedClient =
    new AmazonSQSExtendedClient(sqsClient,
sqsExtendedClientConfiguration);

//Read the message from the queue
final ReceiveMessageResult result =
sqsExtendedClient.receiveMessage(queueUrl);
System.out.println("Received message is " +
result.getMessages().get(0).getBody());
}
}
```

## Cross-service examples for Amazon SNS using AWS SDKs

The following code examples show how to use Amazon Simple Notification Service with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Build an application to submit data to a DynamoDB table \(p. 2015\)](#)
- [Build a publish and subscription application that translates messages \(p. 2016\)](#)
- [Create an Amazon Textract explorer application \(p. 2017\)](#)
- [Detect people and objects in a video with Amazon Rekognition using an AWS SDK \(p. 2018\)](#)
- [Use API Gateway to invoke a Lambda function \(p. 2019\)](#)
- [Use scheduled events to invoke a Lambda function \(p. 2020\)](#)

## Build an application to submit data to a DynamoDB table

The following code examples show how to build an application that submits data to an Amazon DynamoDB table and notifies you when a user updates the table.

Java

### SDK for Java 2.x

Shows how to create a dynamic web application that submits data using the Amazon DynamoDB Java API and sends a text message using the Amazon Simple Notification Service Java API.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon SNS

JavaScript

#### SDK for JavaScript V3

This example shows how to build an app that enables users to submit data to an Amazon DynamoDB table, and send a text message to the administrator using Amazon Simple Notification Service (Amazon SNS).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

### Services used in this example

- DynamoDB
- Amazon SNS

Kotlin

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to create a native Android application that submits data using the Amazon DynamoDB Kotlin API and sends a text message using the Amazon SNS Kotlin API.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon SNS

## Build a publish and subscription application that translates messages

The following code examples show how to create an application that has subscription and publish functionality and translates messages.

.NET

#### AWS SDK for .NET

Shows how to use the Amazon Simple Notification Service .NET API to create a web application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon SNS
- Amazon Translate

Java

#### SDK for Java 2.x

Shows how to use the Amazon Simple Notification Service Java API to create a web application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

For complete source code and instructions on how to set up and run the example that uses the Java Async API, see the full example on [GitHub](#).

#### Services used in this example

- Amazon SNS
- Amazon Translate

Kotlin

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to use the Amazon SNS Kotlin API to create an application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to create a web app, see the full example on [GitHub](#).

For complete source code and instructions on how to create a native Android app, see the full example on [GitHub](#).

#### Services used in this example

- Amazon SNS
- Amazon Translate

## Create an Amazon Textract explorer application

The following code examples show how to explore Amazon Textract output through an interactive application.

JavaScript

#### SDK for JavaScript V3

Shows how to use the AWS SDK for JavaScript to build a React application that uses Amazon Textract to extract data from a document image and display it in an interactive web page.

This example runs in a web browser and requires an authenticated Amazon Cognito identity for credentials. It uses Amazon Simple Storage Service (Amazon S3) for storage, and for notifications it polls an Amazon Simple Queue Service (Amazon SQS) queue that is subscribed to an Amazon Simple Notification Service (Amazon SNS) topic.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

#### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with Amazon Textract to detect text, form, and table elements in a document image. The input image and Amazon Textract output are shown in a Tkinter application that lets you explore the detected elements.

- Submit a document image to Amazon Textract and explore the output of detected elements.
- Submit images directly to Amazon Textract or through an Amazon Simple Storage Service (Amazon S3) bucket.
- Use asynchronous APIs to start a job that publishes a notification to an Amazon Simple Notification Service (Amazon SNS) topic when the job completes.
- Poll an Amazon Simple Queue Service (Amazon SQS) queue for a job completion message and display the results.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

## Detect people and objects in a video with Amazon Rekognition using an AWS SDK

The following code examples show how to detect people and objects in a video with Amazon Rekognition.

Python

#### SDK for Python (Boto3)

Use Amazon Rekognition to detect faces, objects, and people in videos by starting asynchronous detection jobs. This example also configures Amazon Rekognition to notify an Amazon Simple Notification Service (Amazon SNS) topic when jobs complete and subscribes an Amazon Simple

Queue Service (Amazon SQS) queue to the topic. When the queue receives a message about a job, the job is retrieved and the results are output.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon SNS
- Amazon SQS

## Use API Gateway to invoke a Lambda function

The following code examples show how to create an AWS Lambda function invoked by Amazon API Gateway.

Java

### SDK for Java 2.x

Shows how to create an AWS Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create a Lambda function invoked by Amazon API Gateway that scans an Amazon DynamoDB table for work anniversaries and uses Amazon Simple Notification Service (Amazon SNS) to send a text message to your employees that congratulates them at their one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

### SDK for JavaScript V3

Shows how to create an AWS Lambda function by using the Lambda JavaScript runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create a Lambda function invoked by Amazon API Gateway that scans an Amazon DynamoDB table for work anniversaries and uses Amazon Simple Notification Service (Amazon SNS) to send a text message to your employees that congratulates them at their one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

### Services used in this example

- API Gateway
- DynamoDB

- Lambda
- Amazon SNS

## Use scheduled events to invoke a Lambda function

The following code examples show how to create an AWS Lambda function invoked by an Amazon EventBridge scheduled event.

Java

### SDK for Java 2.x

Shows how to create an Amazon EventBridge scheduled event that invokes an AWS Lambda function. Configure EventBridge to use a cron expression to schedule when the Lambda function is invoked. In this example, you create a Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create an app that sends a mobile text message to your employees that congratulates them at the one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

### SDK for JavaScript V3

Shows how to create an Amazon EventBridge scheduled event that invokes an AWS Lambda function. Configure EventBridge to use a cron expression to schedule when the Lambda function is invoked. In this example, you create a Lambda function by using the Lambda JavaScript runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create an app that sends a mobile text message to your employees that congratulates them at the one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

### Services used in this example

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

## Code examples for Amazon SQS using AWS SDKs

The following code examples show how to use Amazon Simple Queue Service with an AWS software development kit (SDK).

## Code examples

- [Actions for Amazon SQS using AWS SDKs \(p. 2021\)](#)
  - [Authorize an Amazon S3 bucket to send messages to an Amazon SQS queue \(p. 2022\)](#)
  - [Change how long an Amazon SQS queue waits for a message \(p. 2022\)](#)
  - [Receive an Amazon SQS message and change its timeout visibility \(p. 2024\)](#)
  - [Create an Amazon SQS queue using an AWS SDK \(p. 2028\)](#)
  - [Delete a batch of messages from an Amazon SQS queue using an AWS SDK \(p. 2039\)](#)
  - [Delete a message from an Amazon SQS queue using an AWS SDK \(p. 2040\)](#)
  - [Delete an Amazon SQS queue using an AWS SDK \(p. 2047\)](#)
  - [Get attributes for an Amazon SQS queue \(p. 2053\)](#)
  - [Get the URL of an Amazon SQS queue using an AWS SDK \(p. 2054\)](#)
  - [List Amazon SQS queues using an AWS SDK \(p. 2059\)](#)
  - [Receive messages from an Amazon SQS queue using an AWS SDK \(p. 2063\)](#)
  - [Send a batch of messages to an Amazon SQS queue using an AWS SDK \(p. 2072\)](#)
  - [Send a message to an Amazon SQS queue using an AWS SDK \(p. 2074\)](#)
- [Scenarios for Amazon SQS using AWS SDKs \(p. 2084\)](#)
  - [Send and receive batches of messages with Amazon SQS using an AWS SDK \(p. 2084\)](#)
- [Cross-service examples for Amazon SQS using AWS SDKs \(p. 2087\)](#)
  - [Create a web application that sends and retrieves messages by using the Amazon Simple Queue Service API \(p. 2087\)](#)
  - [Create a messenger application with Step Functions \(p. 2088\)](#)
  - [Create an Amazon Textract explorer application \(p. 2089\)](#)
  - [Detect people and objects in a video with Amazon Rekognition using an AWS SDK \(p. 2090\)](#)

## Actions for Amazon SQS using AWS SDKs

The following code examples show how to use Amazon Simple Queue Service with AWS SDKs. Each example calls an individual service function.

### Examples

- [Authorize an Amazon S3 bucket to send messages to an Amazon SQS queue \(p. 2022\)](#)
- [Change how long an Amazon SQS queue waits for a message \(p. 2022\)](#)
- [Receive an Amazon SQS message and change its timeout visibility \(p. 2024\)](#)
- [Create an Amazon SQS queue using an AWS SDK \(p. 2028\)](#)
- [Delete a batch of messages from an Amazon SQS queue using an AWS SDK \(p. 2039\)](#)
- [Delete a message from an Amazon SQS queue using an AWS SDK \(p. 2040\)](#)
- [Delete an Amazon SQS queue using an AWS SDK \(p. 2047\)](#)
- [Get attributes for an Amazon SQS queue \(p. 2053\)](#)
- [Get the URL of an Amazon SQS queue using an AWS SDK \(p. 2054\)](#)
- [List Amazon SQS queues using an AWS SDK \(p. 2059\)](#)
- [Receive messages from an Amazon SQS queue using an AWS SDK \(p. 2063\)](#)
- [Send a batch of messages to an Amazon SQS queue using an AWS SDK \(p. 2072\)](#)
- [Send a message to an Amazon SQS queue using an AWS SDK \(p. 2074\)](#)

## Authorize an Amazon S3 bucket to send messages to an Amazon SQS queue

The following code example shows how to authorize an Amazon S3 bucket to send messages to an Amazon SQS queue.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.SQS;

public class AuthorizeS3ToSendMessage
{
    /// <summary>
    /// Initializes the Amazon SQS client object and then calls the
    /// AuthorizeS3ToSendMessageAsync method to authorize the named
    /// bucket to send messages in response to S3 events.
    /// </summary>
    public static async Task Main()
    {
        string queueUrl = "https://sqs.us-east-2.amazonaws.com/0123456789ab/
Example_Queue";
        string bucketName = "doc-example-bucket";

        // Create an Amazon SQS client object using the
        // default user. If the AWS Region you want to use
        // is different, supply the AWS Region as a parameter.
        IAmazonSQS client = new AmazonSQSClient();

        var queueARN = await client.AuthorizeS3ToSendMessageAsync(queueUrl,
bucketName);

        if (!string.IsNullOrEmpty(queueARN))
        {
            Console.WriteLine($"The Amazon S3 bucket: {bucketName} has been
successfully authorized.");
            Console.WriteLine($"{bucketName} can now send messages to the queue
with ARN: {queueARN}.");
        }
    }
}
```

## Change how long an Amazon SQS queue waits for a message

The following code example shows how to change how long an Amazon SQS queue waits for a message to arrive.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Change how long an Amazon SQS queue waits for a message to arrive.

```
// Import required AWS SDK clients and commands for Node.js
import { SetQueueAttributesCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = {
  Attributes: {
    ReceiveMessageWaitTimeSeconds: "20",
  },
  QueueUrl: "SQS_QUEUE_URL", //SQS_QUEUE_URL; e.g., 'https://
  sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'
};

const run = async () => {
  try {
    const data = await sqsClient.send(new SetQueueAttributesCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.error(err, err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SetQueueAttributes](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Change how long an Amazon SQS queue waits for a message to arrive.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the SQS service object
```

```
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {
  Attributes: [
    "ReceiveMessageWaitTimeSeconds": "20",
  ],
  QueueUrl: "SQS_QUEUE_URL"
};

sqs.setQueueAttributes(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SetQueueAttributes](#) in *AWS SDK for JavaScript API Reference*.

## Receive an Amazon SQS message and change its timeout visibility

The following code examples show how to change an Amazon SQS message timeout visibility.

Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
  "context"
  "flag"
  "fmt"
  "strconv"

  "github.com/aws/aws-sdk-go-v2/config"
  "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// SQSSetMsgVisibilityAPI defines the interface for the GetQueueUrl and
// ChangeMessageVisibility functions.
// We use this interface to test the functions using a mocked service.
type SQSSetMsgVisibilityAPI interface {
  GetQueueUrl(ctx context.Context,
    params *sqs.GetQueueUrlInput,
    optFns ...func(*sqs.Options)) (*sqs.GetQueueUrlOutput, error)

  ChangeMessageVisibility(ctx context.Context,
    params *sqs.ChangeMessageVisibilityInput,
    optFns ...func(*sqs.Options)) (*sqs.ChangeMessageVisibilityOutput, error)
}

// GetQueueURL gets the URL of an Amazon SQS queue.
```

```
// Inputs:  
//   c is the context of the method call, which includes the AWS Region.  
//   api is the interface that defines the method call.  
//   input defines the input arguments to the service call.  
// Output:  
//   If success, a GetQueueUrlOutput object containing the result of the service  
//   call and nil.  
//   Otherwise, nil and an error from the call to GetQueueUrl.  
func GetQueueURL(c context.Context, api SQSSetMsgVisibilityAPI, input  
  *sqc.GetQueueUrlInput) (*sqc.GetQueueUrlOutput, error) {  
    return api.GetQueueUrl(c, input)  
  
}  
  
// SetMsgVisibility sets the visibility timeout for a message in an Amazon SQS  
// queue.  
// Inputs:  
//   c is the context of the method call, which includes the AWS Region.  
//   api is the interface that defines the method call.  
//   input defines the input arguments to the service call.  
// Output:  
//   If success, a ChangeMessageVisibilityOutput object containing the result of  
//   the service call and nil.  
//   Otherwise, nil and an error from the call to ChangeMessageVisibility.  
func SetMsgVisibility(c context.Context, api SQSSetMsgVisibilityAPI, input  
  *sqc.ChangeMessageVisibilityInput) (*sqc.ChangeMessageVisibilityOutput, error) {  
    return api.ChangeMessageVisibility(c, input)  
}  
  
func main() {  
    queue := flag.String("q", "", "The name of the queue")  
    handle := flag.String("h", "", "The receipt handle of the message")  
    visibility := flag.Int("v", 30, "The duration, in seconds, that the message is not  
    visible to other consumers")  
    flag.Parse()  
  
    if *queue == "" || *handle == "" {  
        fmt.Println("You must supply a queue name (-q QUEUE) and message receipt handle  
        (-h HANDLE)")  
        return  
    }  
  
    if *visibility < 0 {  
        *visibility = 0  
    }  
  
    if *visibility > 12*60*60 {  
        *visibility = 12 * 60 * 60  
    }  
  
    cfg, err := config.LoadDefaultConfig(context.TODO())  
    if err != nil {  
        panic("configuration error, " + err.Error())  
    }  
  
    client := sqs.NewFromConfig(cfg)  
  
    gQInput := &sqc.GetQueueUrlInput{  
        QueueName: queue,  
    }  
  
    // Get URL of queue  
    urlResult, err := GetQueueURL(context.TODO(), client, gQInput)  
    if err != nil {  
        fmt.Println("Got an error getting the queue URL:")
```

```
    fmt.Println(err)
    return
}

queueURL := urlResult.QueueUrl

sVInput := &sqs.ChangeMessageVisibilityInput{
    ReceiptHandle:    handle,
    QueueUrl:        queueURL,
    VisibilityTimeout: int32(*visibility),
}

_, err = SetMsgVisibility(context.TODO(), client, sVInput)
if err != nil {
    fmt.Println("Got an error setting the visibility of the message:")
    fmt.Println(err)
    return
}

fmt.Println("Changed the visibility of the message with the handle " + *handle + " in the " + *queue + " to " + strconv.Itoa(*visibility))
}
```

- For API details, see [ChangeMessageVisibility](#) in *AWS SDK for Go API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Receive an Amazon SQS message and change its timeout visibility.

```
// Import required AWS SDK clients and commands for Node.js
import {
  ReceiveMessageCommand,
  ChangeMessageVisibilityCommand,
} from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const queueURL = "https://sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME"; //
REGION, ACCOUNT_ID, QUEUE_NAME
const params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
```

```
};

const run = async () => {
  try {
    const data = await sqsClient.send(new ReceiveMessageCommand(params));
    if (data.Messages != null) {
      try {
        const visibilityParams = {
          QueueUrl: queueURL,
          ReceiptHandle: data.Messages[0].ReceiptHandle,
          VisibilityTimeout: 20, // 20 second timeout
        };
        const results = await sqsClient.send(
          new ChangeMessageVisibilityCommand(visibilityParams)
        );
        console.log("Timeout Changed", results);
      } catch (err) {
        console.log("Delete Error", err);
      }
    } else {
      console.log("No messages to change");
    }
    return data; // For unit tests.
  } catch (err) {
    console.log("Receive Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ChangeMessageVisibility](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Receive an Amazon SQS message and change its timeout visibility.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk')
// Set the region to us-west-2
AWS.config.update({ region: 'us-west-2' })

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: '2012-11-05' })

var queueURL = 'https://sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'

var params = {
  AttributeNames: ['SentTimestamp'],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ['All'],
  QueueUrl: queueURL
}

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log('Receive Error', err)
  } else {
    // Make sure we have a message
    if (data.Messages != null) {
```

```
var visibilityParams = {
  QueueUrl: queueURL,
  ReceiptHandle: data.Messages[0].ReceiptHandle,
  VisibilityTimeout: 20 // 20 second timeout
}
sqS.changeMessageVisibility(visibilityParams, function (err, data) {
  if (err) {
    console.log('Delete Error', err)
  } else {
    console.log('Timeout Changed', data)
  }
})
} else {
  console.log('No messages to change')
}
})
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ChangeMessageVisibility](#) in [AWS SDK for JavaScript API Reference](#).

## Create an Amazon SQS queue using an AWS SDK

The following code examples show how to create an Amazon SQS queue.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SQS;
using Amazon.SQS.Model;

public class CreateQueue
{
  /// <summary>
  /// Initializes the Amazon SQS client object and then calls the
  /// CreateQueueAsync method to create the new queue. If the call is
  /// successful, it displays the URL of the new queue on the console.
  /// </summary>
  public static async Task Main()
  {
    // If the Amazon SQS message queue is not in the same AWS Region as
    // your default user, you need to provide the AWS Region as a parameter to
    // the client constructor.
    var client = new AmazonSQSClient();

    string queueName = "New-Example-Queue";
    int maxMessage = 256 * 1024;
    var attrs = new Dictionary<string, string>
    {
      {
```

```
        QueueAttributeName.DelaySeconds,
        TimeSpan.FromSeconds(5).TotalSeconds.ToString()
    },
    {
        QueueAttributeName.MaximumMessageSize,
        maxMessage.ToString()
    },
    {
        QueueAttributeName.MessageRetentionPeriod,
        TimeSpan.FromDays(4).TotalSeconds.ToString()
    },
    {
        QueueAttributeName.ReceiveMessageWaitTimeSeconds,
        TimeSpan.FromSeconds(5).TotalSeconds.ToString()
    },
    {
        QueueAttributeName.VisibilityTimeout,
        TimeSpan.FromHours(12).TotalSeconds.ToString()
    },
};

var request = new CreateQueueRequest
{
    Attributes = attrs,
    QueueName = queueName,
};

var response = await client.CreateQueueAsync(request);

if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
{
    Console.WriteLine("Successfully created Amazon SQS queue.");
    Console.WriteLine($"Queue URL: {response.QueueUrl}");
}
}
```

Create an Amazon SQS queue and send a message to it.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.SQS;
using Amazon.SQS.Model;

public class CreateSendExample
{
    // Specify your AWS Region (an example Region is shown).
    private static readonly string QueueName = "Example_Queue";
    private static readonly RegionEndpoint ServiceRegion =
RegionEndpoint.USWest2;
    private static IAmazonSQS client;

    public static async Task Main()
    {
        client = new AmazonSQSClient(ServiceRegion);
        var createQueueResponse = await CreateQueue(client, QueueName);

        string queueUrl = createQueueResponse.QueueUrl;

        Dictionary<string, MessageAttributeValue> messageAttributes = new
Dictionary<string, MessageAttributeValue>
```

```

        {
            { "Title", new MessageAttributeValue { DataType = "String",
StringValue = "The Whistler" } },
            { "Author", new MessageAttributeValue { DataType = "String",
StringValue = "John Grisham" } },
            { "WeeksOn", new MessageAttributeValue { DataType = "Number",
StringValue = "6" } };
        };

        string messageBody = "Information about current NY Times fiction
bestseller for week of 12/11/2016./";

        var sendMsgResponse = await SendMessage(client, queueUrl, messageBody,
messageAttributes);
    }

    /// <summary>
    /// Creates a new Amazon SQS queue using the queue name passed to it
    /// in queueName.
    /// </summary>
    /// <param name="client">An SQS client object used to send the message.</
param>
    /// <param name="queueName">A string representing the name of the queue
    /// to create.</param>
    /// <returns>A CreateQueueResponse that contains information about the
    /// newly created queue.</returns>
    public static async Task<CreateQueueResponse> CreateQueue(IAmazonSQS
client, string queueName)
{
    var request = new CreateQueueRequest
    {
        QueueName = queueName,
        Attributes = new Dictionary<string, string>
        {
            { "DelaySeconds", "60" },
            { "MessageRetentionPeriod", "86400" },
        },
    };

    var response = await client.CreateQueueAsync(request);
    Console.WriteLine($"Created a queue with URL : {response.QueueUrl}");

    return response;
}

    /// <summary>
    /// Sends a message to an SQS queue.
    /// </summary>
    /// <param name="client">An SQS client object used to send the message.</
param>
    /// <param name="queueUrl">The URL of the queue to which to send the
    /// message.</param>
    /// <param name="messageBody">A string representing the body of the
    /// message to be sent to the queue.</param>
    /// <param name="messageAttributes">Attributes for the message to be
    /// sent to the queue.</param>
    /// <returns>A SendMessageResponse object that contains information
    /// about the message that was sent.</returns>
    public static async Task<SendMessageResponse> SendMessage(
        IAmazonSQS client,
        string queueUrl,
        string messageBody,
        Dictionary<string, MessageAttributeValue> messageAttributes)
{
    var sendMessageRequest = new SendMessageRequest
    {

```

```
        DelaySeconds = 10,
        MessageAttributes = messageAttributes,
        MessageBody = messageBody,
        QueueUrl = queueUrl,
    };

    var response = await client.SendMessageAsync(sendMessageRequest);
    Console.WriteLine($"Sent a message with id : {response.MessageId}");

    return response;
}
}
```

- For API details, see [CreateQueue](#) in *AWS SDK for .NET API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// SQSCreateQueueAPI defines the interface for the CreateQueue function.
// We use this interface to test the function using a mocked service.
type SQSCreateQueueAPI interface {
    CreateQueue(ctx context.Context,
        params *sqs.CreateQueueInput,
        optFns ...func(*sqs.Options)) (*sqs.CreateQueueOutput, error)
}

// CreateQueue creates an Amazon SQS queue.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If success, a CreateQueueOutput object containing the result of the service
//     call and nil.
//     Otherwise, nil and an error from the call to CreateQueue.
func CreateQueue(c context.Context, api SQSCreateQueueAPI, input
    *sqs.CreateQueueInput) (*sqs.CreateQueueOutput, error) {
    return api.CreateQueue(c, input)
}

func main() {
    queue := flag.String("q", "", "The name of the queue")
    flag.Parse()

    if *queue == "" {
```

```
    fmt.Println("You must supply a queue name (-q QUEUE")
    return
}

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic("configuration error, " + err.Error())
}

client := sqs.NewFromConfig(cfg)

input := &sqs.CreateQueueInput{
    QueueName: queue,
    Attributes: map[string]string{
        "DelaySeconds": "60",
        "MessageRetentionPeriod": "86400",
    },
}

result, err := CreateQueue(context.TODO(), client, input)
if err != nil {
    fmt.Println("Got an error creating the queue:")
    fmt.Println(err)
    return
}

fmt.Println("URL: " + *result.QueueUrl)
}
```

- For API details, see [CreateQueue](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createQueue(SqsClient sqsClient, String queueName ) {

    try {
        System.out.println("\nCreate Queue");

        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
            .queueName(queueName)
            .build();

        sqsClient.createQueue(createQueueRequest);

        System.out.println("\nGet queue url");

        GetQueueUrlResponse getQueueUrlResponse =
            sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        return getQueueUrlResponse.queueUrl();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

```

}

public static void listQueues(SqsClient sqsClient) {

    System.out.println("\nList Queues");
    String prefix = "que";

    try {
        ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
        ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
        for (String url : listQueuesResponse.queueUrls()) {
            System.out.println(url);
        }
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listQueuesFilter(SqsClient sqsClient, String queueUrl ) {
    // List queues with filters
    String namePrefix = "queue";
    ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
        .queueNamePrefix(namePrefix)
        .build();

    ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
    System.out.println("Queue URLs with prefix: " + namePrefix);
    for (String url : listQueuesFilteredResponse.queueUrls()) {
        System.out.println(url);
    }

    System.out.println("\nSend message");
    try {
        sqsClient.sendMessage.SendMessageRequest.builder()
            .queueUrl(queueUrl)
            .messageBody("Hello world!")
            .delaySeconds(10)
            .build();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

    System.out.println("\nSend multiple messages");
    try {
        SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
            .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10).build())
            .build();
        sqsClient.sendMessageBatch(sendMessageBatchRequest);
    }
}

```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String queueUrl) {
        System.out.println("\nReceive messages");
        try {
            ReceiveMessageRequest receiveMessageRequest =
                ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void changeMessages(SqsClient sqsClient, String queueUrl,
        List<Message> messages) {
        System.out.println("\nChange Message Visibility");
        try {
            for (Message message : messages) {
                ChangeMessageVisibilityRequest req =
                    ChangeMessageVisibilityRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .visibilityTimeout(100)
                        .build();
                sqsClient.changeMessageVisibility(req);
            }
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
        List<Message> messages) {
        System.out.println("\nDelete Messages");

        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
                    DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [CreateQueue](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Create an Amazon SQS standard queue.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateQueueCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = {
  QueueName: "SQS_QUEUE_NAME", //SQS_QUEUE_URL
  Attributes: {
    DelaySeconds: "60", // Number of seconds delay.
    MessageRetentionPeriod: "86400", // Number of seconds delay.
  },
};

const run = async () => {
  try {
    const data = await sqsClient.send(new CreateQueueCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Create an Amazon SQS queue that waits for a message to arrive.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateQueueCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = {
  QueueName: "SQS_QUEUE_NAME", //SQS_QUEUE_URL; e.g., 'https://
  sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'
  Attributes: {
    ReceiveMessageWaitTimeSeconds: "20",
```

```
        },
    };

const run = async () => {
    try {
        const data = await sqsClient.send(new CreateQueueCommand(params));
        console.log("Success", data);
        return data; // For unit tests.
    } catch (err) {
        console.error(err, err.stack);
    }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateQueue](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an Amazon SQS standard queue.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {
    QueueName: 'SQS_QUEUE_NAME',
    Attributes: {
        'DelaySeconds': '60',
        'MessageRetentionPeriod': '86400'
    }
};

sqs.createQueue(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data.QueueUrl);
    }
});
```

Create an Amazon SQS queue that waits for a message to arrive.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {
    QueueName: 'SQS_QUEUE_NAME',
```

```
    Attributes: {
      'ReceiveMessageWaitTimeSeconds': '20',
    }
  };

  sqs.createQueue(params, function(err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data.QueueUrl);
    }
  });
}
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateQueue](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createQueue(queueNameVal: String): String {

  println("Create Queue")
  val createQueueRequest = CreateQueueRequest {
    queueName = queueNameVal
  }

  SqsClient { region = "us-east-1" }.use { sqsClient ->
    sqsClient.createQueue(createQueueRequest)
    println("Get queue url")

    val getQueueUrlRequest = GetQueueUrlRequest {
      queueName = queueNameVal
    }

    val getQueueUrlResponse = sqsClient.getQueueUrl(getQueueUrlRequest)
    return getQueueUrlResponse.queueUrl.toString()
  }
}
```

- For API details, see [CreateQueue](#) in [AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_queue(name, attributes=None):
    """
    Creates an Amazon SQS queue.

    :param name: The name of the queue. This is part of the URL assigned to the
    queue.
    :param attributes: The attributes of the queue, such as maximum message size or
        whether it's a FIFO queue.
    :return: A Queue object that contains metadata about the queue and that can be
    used
        to perform queue operations like sending and receiving messages.
    """
    if not attributes:
        attributes = {}

    try:
        queue = sqs.create_queue(
            QueueName=name,
            Attributes=attributes
        )
        logger.info("Created queue '%s' with URL=%s", name, queue.url)
    except ClientError as error:
        logger.exception("Couldn't create queue named '%s'.", name)
        raise error
    else:
        return queue
```

- For API details, see [CreateQueue](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an Amazon SQS standard queue.

```
TRY.
    oo_result = lo_sqs->createqueue( iv_queuename = iv_queue_name ).          "
oo_result is returned for testing purpose "
    MESSAGE 'SQS queue created' TYPE 'I'.
    CATCH /aws1/cx_sqssqueuedeldrecently.
    MESSAGE 'After deleting a queue, you should wait 60 seconds before creating
another queue with the same name.' TYPE 'E'.
    CATCH /aws1/cx_sqssqueuenameexists.
    MESSAGE 'A queue with this name already exists.' TYPE 'E'.
ENDTRY.
```

Create an Amazon SQS queue that waits for a message to arrive.

```
TRY.
    DATA lt_attributes TYPE /aws1/cl_sqssqueueattrmap_w=>tt_queueattributemap.
    DATA ls_attribute TYPE /aws1/
cl_sqssqueueattrmap_w=>ts_queueattributemap_maprow.
```

```
    ls_attribute-key = 'ReceiveMessageWaitTimeSeconds'.           " time
in seconds for long polling i.e. time for which the call waits for a message to
arrive in queue before returning "
    ls_attribute-value = NEW /aws1/cl_sqssqueueattrmap_w( iv_value =
iv_wait_time ).
    INSERT ls_attribute INTO TABLE lt_attributes.
    oo_result = lo_sqe->createqueue(                               " oo_result is returned
for testing purpose "
        iv_queueusername = iv_queue_name
        it_attributes = lt_attributes
    ).
MESSAGE 'SQS queue created' TYPE 'I'.
CATCH /aws1/cx_sqssqueuedeldrecently.
MESSAGE 'After deleting a queue, you should wait 60 seconds before creating
another queue with the same name.' TYPE 'E'.
CATCH /aws1/cx_sqssqueuenameexists.
MESSAGE 'A queue with this name already exists.' TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateQueue](#) in *AWS SDK for SAP ABAP API reference*.

## Delete a batch of messages from an Amazon SQS queue using an AWS SDK

The following code example shows how to delete a batch of messages from an Amazon SQS queue.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_messages(queue, messages):
    """
    Delete a batch of messages from a queue in a single request.

    :param queue: The queue from which to delete the messages.
    :param messages: The list of messages to delete.
    :return: The response from SQS that contains the list of successful and failed
            message deletions.
    """
    try:
        entries = [
            {
                'Id': str(ind),
                'ReceiptHandle': msg.receipt_handle
            } for ind, msg in enumerate(messages)]
        response = queue.delete_messages(Entries=entries)
        if 'Successful' in response:
            for msg_meta in response['Successful']:
                logger.info("Deleted %s",
                            messages[int(msg_meta['Id'])].receipt_handle)
        if 'Failed' in response:
            for msg_meta in response['Failed']:
                logger.warning(
                    "Could not delete %s",
                    messages[int(msg_meta['Id'])].receipt_handle
                )
    except ClientError:
```

```
    logger.exception("Couldn't delete messages from queue %s", queue)
else:
    return response
```

- For API details, see [DeleteMessageBatch](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a message from an Amazon SQS queue using an AWS SDK

The following code examples show how to delete a message from an Amazon SQS queue.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SQS;
using Amazon.SQS.Model;

public class DeleteMessage
{
    /// <summary>
    /// Initializes the Amazon SQS client object. It then calls the
    /// ReceiveMessageAsync method to retrieve information about the
    /// available methods before deleting them.
    /// </summary>
    public static async Task Main()
    {
        string queueUrl = "https://sqs.us-east-2.amazonaws.com/0123456789ab/
Example_Queue";
        var attributeNames = new List<string>() { "All" };
        int maxNumberOfMessages = 5;
        var visibilityTimeout = (int)TimeSpan.FromMinutes(10).TotalSeconds;
        var waitTimeSeconds = (int)TimeSpan.FromSeconds(5).TotalSeconds;

        // If the Amazon SQS message queue is not in the same AWS Region as
        // your
        // default user, you need to provide the AWS Region as a parameter to
        // the
        // client constructor.
        var client = new AmazonSQSClient();

        var request = new ReceiveMessageRequest
        {
            QueueUrl = queueUrl,
            AttributeNames = attributeNames,
            MaxNumberOfMessages = maxNumberOfMessages,
            VisibilityTimeout = visibilityTimeout,
            WaitTimeSeconds = waitTimeSeconds,
        };

        var response = await client.ReceiveMessageAsync(request);

        if (response.Messages.Count > 0)
```

```

    {
        response.Messages.ForEach(async m =>
        {
            Console.WriteLine($"Message ID: '{m.MessageId}'");

            var delRequest = new DeleteMessageRequest
            {
                QueueUrl = "https://sqs.us-
east-1.amazonaws.com/0123456789ab/MyTestQueue",
                ReceiptHandle = m.ReceiptHandle,
            };

            var delResponse = await client.DeleteMessageAsync(delRequest);
        });
    }
    else
    {
        Console.WriteLine("No messages to delete.");
    }
}
}

```

Receive a message from an Amazon SQS queue and then delete the message.

```

public static async Task Main()
{
    // If the AWS Region you want to use is different from
    // the AWS Region defined for the default user, supply
    // the specify your AWS Region to the client constructor.
    var client = new AmazonSQSClient();
    string queueName = "Example_Queue";

    var queueUrl = await GetQueueUrl(client, queueName);
    Console.WriteLine($"The SQS queue's URL is {queueUrl}");

    var response = await ReceiveAndDeleteMessage(client, queueUrl);

    Console.WriteLine($"Message: {response.Messages[0]}");
}

/// <summary>
/// Retrieve the queue URL for the queue named in the queueName
/// property using the client object.
/// </summary>
/// <param name="client">The Amazon SQS client used to retrieve the
/// queue URL.</param>
/// <param name="queueName">A string representing name of the queue
/// for which to retrieve the URL.</param>
/// <returns>The URL of the queue.</returns>
public static async Task<string> GetQueueUrl(IAmazonSQS client, string
queueName)
{
    var request = new GetQueueUrlRequest
    {
        QueueName = queueName,
    };

    GetQueueUrlResponse response = await client.GetQueueUrlAsync(request);
    return response.QueueUrl;
}

/// <summary>
/// Retrieves the message from the queue at the URL passed in the

```

```
    ///> queueURL parameters using the client.  
    ///> </summary>  
    ///> <param name="client">The SQS client used to retrieve a message.</param>  
    ///> <param name="queueUrl">The URL of the queue from which to retrieve  
    ///> a message.</param>  
    ///> <returns>The response from the call to ReceiveMessageAsync.</returns>  
    public static async Task<ReceiveMessageResponse>  
        ReceiveAndDeleteMessage(IAmazonSQS client, string queueUrl)  
    {  
        // Receive a single message from the queue.  
        var receiveMessageRequest = new ReceiveMessageRequest  
        {  
            AttributeNames = { "SentTimestamp" },  
            MaxNumberOfMessages = 1,  
            MessageAttributeNames = { "All" },  
            QueueUrl = queueUrl,  
            VisibilityTimeout = 0,  
            WaitTimeSeconds = 0,  
        };  
  
        var receiveMessageResponse = await  
            client.ReceiveMessageAsync(receiveMessageRequest);  
  
        // Delete the received message from the queue.  
        var deleteMessageRequest = new DeleteMessageRequest  
        {  
            QueueUrl = queueUrl,  
            ReceiptHandle = receiveMessageResponse.Messages[0].ReceiptHandle,  
        };  
  
        await client.DeleteMessageAsync(deleteMessageRequest);  
  
        return receiveMessageResponse;  
    }  
}
```

- For API details, see [DeleteMessage](#) in *AWS SDK for .NET API Reference*.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main  
  
import (  
    "context"  
    "flag"  
    "fmt"  
  
    "github.com/aws/aws-sdk-go-v2/config"  
    "github.com/aws/aws-sdk-go-v2/service/sqs"  
)  
  
// SQSDelteMessageAPI defines the interface for the GetQueueUrl and DeleteMessage  
// functions.  
// We use this interface to test the functions using a mocked service.  
type SQSDelteMessageAPI interface {
```

```
GetQueueUrl(ctx context.Context,
    params *sqsc.GetQueueUrlInput,
    optFns ...func(*sqsc.Options)) (*sqsc.GetQueueUrlOutput, error)

DeleteMessage(ctx context.Context,
    params *sqsc.DeleteMessageInput,
    optFns ...func(*sqsc.Options)) (*sqsc.DeleteMessageOutput, error)
}

// GetQueueURL gets the URL of an Amazon SQS queue.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If success, a GetQueueUrlOutput object containing the result of the service
//   call and nil.
//   Otherwise, nil and an error from the call to GetQueueUrl.
func GetQueueURL(c context.Context, api SQSDelateMessageAPI, input
    *sqsc.GetQueueUrlInput) (*sqsc.GetQueueUrlOutput, error) {
    return api.GetQueueUrl(c, input)
}

// RemoveMessage deletes a message from an Amazon SQS queue.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If success, a DeleteMessageOutput object containing the result of the
//   service call and nil.
//   Otherwise, nil and an error from the call to DeleteMessage.
func RemoveMessage(c context.Context, api SQSDelateMessageAPI, input
    *sqsc.DeleteMessageInput) (*sqsc.DeleteMessageOutput, error) {
    return api.DeleteMessage(c, input)
}

func main() {
    queue := flag.String("q", "", "The name of the queue")
    messageHandle := flag.String("m", "", "The receipt handle of the message")
    flag.Parse()

    if *queue == "" || *messageHandle == "" {
        fmt.Println("You must supply a queue name (-q QUEUE) and message receipt handle"
        " (-m MESSAGE-HANDLE)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := sqs.NewFromConfig(cfg)

    qUINput := &sqsc.GetQueueUrlInput{
        QueueName: queue,
    }

    // Get URL of queue
    result, err := GetQueueURL(context.TODO(), client, qUINput)
    if err != nil {
        fmt.Println("Got an error getting the queue URL:")
        fmt.Println(err)
        return
    }
}
```

```
queueURL := result.QueueUrl

dMInput := &sqs.DeleteMessageInput{
    QueueUrl:      queueURL,
    ReceiptHandle: messageHandle,
}

_, err = RemoveMessage(context.TODO(), client, dMInput)
if err != nil {
    fmt.Println("Got an error deleting the message:")
    fmt.Println(err)
    return
}

fmt.Println("Deleted message from queue with URL " + *queueURL)
```

- For API details, see [DeleteMessage](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
    .queueUrl(queueUrl)
    .receiptHandle(message.receiptHandle())
    .build();
    sqsClient.deleteMessage(deleteMessageRequest);
}
```

- For API details, see [DeleteMessage](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Receive and delete Amazon SQS messages.

```
// Import required AWS SDK clients and commands for Node.js
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
} from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const queueURL = "SQS_QUEUE_URL"; //SQS_QUEUE_URL; e.g., 'https://
sq.s.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'
const params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 10,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  VisibilityTimeout: 20,
  WaitTimeSeconds: 0,
};

const run = async () => {
  try {
    const data = await sqsClient.send(new ReceiveMessageCommand(params));
    if (data.Messages) {
      var deleteParams = {
        QueueUrl: queueURL,
        ReceiptHandle: data.Messages[0].ReceiptHandle,
      };
      try {
        const data = await sqsClient.send(new DeleteMessageCommand(deleteParams));
        console.log("Message deleted", data);
      } catch (err) {
        console.log("Error", err);
      }
    } else {
      console.log("No messages to delete");
    }
    return data; // For unit tests.
  } catch (err) {
    console.log("Receive Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteMessage](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Receive and delete Amazon SQS messages.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});
```

```
// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var queueURL = "SQS_QUEUE_URL";

var params = {
  AttributeNames: [
    "SentTimestamp"
  ],
  MaxNumberOfMessages: 10,
  MessageAttributeNames: [
    "All"
  ],
  QueueUrl: queueURL,
  VisibilityTimeout: 20,
  WaitTimeSeconds: 0
};

sqs.receiveMessage(params, function(err, data) {
  if (err) {
    console.log("Receive Error", err);
  } else if (data.Messages) {
    var deleteParams = {
      QueueUrl: queueURL,
      ReceiptHandle: data.Messages[0].ReceiptHandle
    };
    sqs.deleteMessage(deleteParams, function(err, data) {
      if (err) {
        console.log("Delete Error", err);
      } else {
        console.log("Message Deleted", data);
      }
    });
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteMessage](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteMessages(queueUrlVal: String) {
  println("Delete Messages from $queueUrlVal")

  val purgeRequest = PurgeQueueRequest {
    queueUrl = queueUrlVal
  }

  SqsClient { region = "us-east-1" }.use { sqsClient ->
    sqsClient.purgeQueue(purgeRequest)
  }
}
```

```
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {
    val request = DeleteQueueRequest {
        queueUrl = queueUrlVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- For API details, see [DeleteMessage](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_message(message):
    """
    Delete a message from a queue. Clients must delete messages after they
    are received and processed to remove them from the queue.

    :param message: The message to delete. The message's queue URL is contained in
                    the message's metadata.
    :return: None
    """
    try:
        message.delete()
        logger.info("Deleted message: %s", message.message_id)
    except ClientError as error:
        logger.exception("Couldn't delete message: %s", message.message_id)
        raise error
```

- For API details, see [DeleteMessage](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an Amazon SQS queue using an AWS SDK

The following code examples show how to delete an Amazon SQS queue.

### .NET

#### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
```

```
using System.Threading.Tasks;
using Amazon.SQS;

public class DeleteQueue
{
    /// <summary>
    /// Initializes the Amazon SQS client object and then calls the
    /// DeleteQueueAsync method to delete the queue.
    /// </summary>
    public static async Task Main()
    {
        // If the Amazon SQS message queue is not in the same AWS Region as
        // default user, you need to provide the AWS Region as a parameter to
        // client constructor.
        var client = new AmazonSQSClient();

        string queueUrl = "https://sqs.us-east-2.amazonaws.com/0123456789ab/
New-Example-Queue";

        var response = await client.DeleteQueueAsync(queueUrl);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("Successfully deleted the queue.");
        }
        else
        {
            Console.WriteLine("Could not delete the queue.");
        }
    }
}
```

- For API details, see [DeleteQueue](#) in *AWS SDK for .NET API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an Amazon SQS queue.

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// SQSDelteQueueAPI defines the interface for the GetQueueUrl and DeleteQueue
// functions.
// We use this interface to test the functions using a mocked service.
type SQSDelteQueueAPI interface {
    GetQueueUrl(ctx context.Context,
```

```
    params *sqc.GetQueueUrlInput,
    optFns ...func(*sqc.Options)) (*sqc.GetQueueUrlOutput, error)

    DeleteQueue(ctx context.Context,
        params *sqc.DeleteQueueInput,
        optFns ...func(*sqc.Options)) (*sqc.DeleteQueueOutput, error)
    }

    // GetQueueURL gets the URL of an Amazon SQS queue.
    // Inputs:
    //   c is the context of the method call, which includes the AWS Region.
    //   api is the interface that defines the method call.
    //   input defines the input arguments to the service call.
    // Output:
    //   If success, a GetQueueUrlOutput object containing the result of the service
    //   call and nil.
    //   Otherwise, nil and an error from the call to GetQueueUrl.
    func GetQueueURL(c context.Context, api SQSDelQueueAPI, input
        *sqc.GetQueueUrlInput) (*sqc.GetQueueUrlOutput, error) {
        return api.GetQueueUrl(c, input)
    }

    // DeleteQueue deletes an Amazon SQS queue.
    // Inputs:
    //   c is the context of the method call, which includes the AWS Region.
    //   api is the interface that defines the method call.
    //   input defines the input arguments to the service call.
    // Output:
    //   If success, a DeleteQueueOutput object containing the result of the service
    //   call and nil.
    //   Otherwise, nil and an error from the call to DeleteQueue.
    func DeleteQueue(c context.Context, api SQSDelQueueAPI, input
        *sqc.DeleteQueueInput) (*sqc.DeleteQueueOutput, error) {
        return api.DeleteQueue(c, input)
    }

    func main() {
        queue := flag.String("q", "", "The name of the queue")
        flag.Parse()

        if *queue == "" {
            fmt.Println("You must supply a queue name (-q QUEUE")
            return
        }

        cfg, err := config.LoadDefaultConfig(context.TODO())
        if err != nil {
            panic("configuration error, " + err.Error())
        }

        client := sqs.NewFromConfig(cfg)

        qInput := &sqc.GetQueueUrlInput{
            QueueName: queue,
        }

        // Get the URL for the queue
        result, err := GetQueueURL(context.TODO(), client, qInput)
        if err != nil {
            fmt.Println("Got an error getting the queue URL:")
            fmt.Println(err)
            return
        }

        queueURL := result.QueueUrl
```

```
dqInput := &sqs.DeleteQueueInput{
    QueueUrl: queueURL,
}

_, err = DeleteQueue(context.TODO(), client, dqInput)
if err != nil {
    fmt.Println("Got an error deleting the queue:")
    fmt.Println(err)
    return
}

fmt.Println("Deleted queue with URL " + *queueURL)
```

- For API details, see [DeleteQueue](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {

    try {
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteQueue](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
```

```
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create SQS service object.  
const sqsClient = new SQSClient({ region: REGION });  
export { sqsClient };
```

Delete an Amazon SQS queue.

```
// Import required AWS SDK clients and commands for Node.js  
import { DeleteQueueCommand } from "@aws-sdk/client-sqs";  
import { sqsClient } from "./libs/sqsClient.js";  
  
// Set the parameters  
const params = { QueueUrl: "SQS_QUEUE_URL" }; //SQS_QUEUE_URL e.g., 'https://  
sqS.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'  
  
const run = async () => {  
    try {  
        const data = await sqsClient.send(new DeleteQueueCommand(params));  
        console.log("Success", data);  
        return data; // For unit tests.  
    } catch (err) {  
        console.error(err, err.stack);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteQueue](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an Amazon SQS queue.

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create an SQS service object  
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});  
  
var params = {  
    QueueUrl: 'SQS_QUEUE_URL'  
};  
  
sqs.deleteQueue(params, function(err, data) {  
    if (err) {  
        console.log("Error", err);  
    } else {  
        console.log("Success", data);  
    }  
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteQueue](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteMessages(queueUrlVal: String) {
    println("Delete Messages from $queueUrlVal")

    val purgeRequest = PurgeQueueRequest {
        queueUrl = queueUrlVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {

    val request = DeleteQueueRequest {
        queueUrl = queueUrlVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- For API details, see [DeleteQueue](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def remove_queue(queue):
    """
    Removes an SQS queue. When run against an AWS account, it can take up to
    60 seconds before the queue is actually deleted.

    :param queue: The queue to delete.
    :return: None
    """
    try:
        queue.delete()
        logger.info("Deleted queue with URL=%s.", queue.url)
    except ClientError as error:
        logger.exception("Couldn't delete queue with URL=%s!", queue.url)
        raise error
```

- For API details, see [DeleteQueue](#) in *AWS SDK for Python (Boto3) API Reference*.

SAP ABAP

#### **SDK for SAP ABAP**

##### **Note**

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

##### **Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_sqs->deletequeue( iv_queueurl = iv_queue_url ).  
    MESSAGE 'SQS queue deleted' TYPE 'I'.  
ENDTRY.
```

- For API details, see [DeleteQueue](#) in *AWS SDK for SAP ABAP API reference*.

## Get attributes for an Amazon SQS queue

The following code example shows how to get attributes for an Amazon SQS queue.

.NET

#### **AWS SDK for .NET**

##### **Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Collections.Generic;  
using System.Threading.Tasks;  
using Amazon.SQS;  
using Amazon.SQS.Model;  
  
public class GetQueueAttributes  
{  
    ///<summary>  
    /// Initializes the Amazon SQS client and then uses it to call the  
    /// GetQueueAttributesAsync method to retrieve the attributes for the  
    /// Amazon SQS queue.  
    ///</summary>  
    public static async Task Main()  
    {  
        // If the Amazon SQS message queue is not in the same AWS Region as  
        // your  
        // default user, you need to provide the AWS Region as a parameter to  
        // the  
        // client constructor.  
        var client = new AmazonSQSClient();  
  
        var queueUrl = "https://sqs.us-east-2.amazonaws.com/0123456789ab/New-  
Example-Queue";
```

```
var attrs = new List<string>() { "All" };

var request = new GetQueueAttributesRequest
{
    QueueUrl = queueUrl,
    AttributeNames = attrs,
};

var response = await client.GetQueueAttributesAsync(request);

if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
{
    DisplayAttributes(response);
}

/// <summary>
/// Displays the attributes passed to the method on the console.
/// </summary>
/// <param name="attrs">The attributes for the Amazon SQS queue.</param>
public static void DisplayAttributes(GetQueueAttributesResponse attrs)
{
    Console.WriteLine($"Attributes for queue ARN '{attrs.QueueARN}':");
    Console.WriteLine($"  Approximate number of messages: {attrs.ApproximateNumberOfMessages}");
    Console.WriteLine($"  Approximate number of messages delayed: {attrs.ApproximateNumberOfMessagesDelayed}");
    Console.WriteLine($"  Approximate number of messages not visible: {attrs.ApproximateNumberOfMessagesNotVisible}");
    Console.WriteLine($"  Queue created on: {attrs.CreatedTimestamp}");
    Console.WriteLine($"  Delay seconds: {attrs.DelaySeconds}");
    Console.WriteLine($"  Queue last modified on: {attrs.LastModifiedTimestamp}");
    Console.WriteLine($"  Maximum message size: {attrs.MaximumMessageSize}");
    Console.WriteLine($"  Message retention period: {attrs.MessageRetentionPeriod}");
    Console.WriteLine($"  Visibility timeout: {attrs.VisibilityTimeout}");
    Console.WriteLine($"  Policy: {attrs.Policy}\n");
    Console.WriteLine("  Attributes:");

    foreach (var attr in attrs.Attributes)
    {
        Console.WriteLine($"    {attr.Key}: {attr.Value}");
    }
}
```

- For API details, see [GetQueueAttributes](#) in *AWS SDK for .NET API Reference*.

## Get the URL of an Amazon SQS queue using an AWS SDK

The following code examples show how to get the URL of an Amazon SQS queue.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.SQS;
using Amazon.SQS.Model;

public class GetQueueUrl
{
    ///<summary>
    ///<summary>Initializes the Amazon SQS client object and then calls the
    ///</summary>GetQueueUrlAsync method to retrieve the URL of an Amazon SQS
    ///queue.
    ///</summary>
    public static async Task Main()
    {
        // If the Amazon SQS message queue is not in the same AWS Region as
your
        // default user, you need to provide the AWS Region as a parameter to
the
        // client constructor.
        var client = new AmazonSQSClient();

        string queueName = "New-Exampe-Queue";

        try
        {
            var response = await client.GetQueueUrlAsync(queueName);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"The URL for {queueName} is:
{response.QueueUrl}");
            }
        }
        catch (QueueDoesNotExistException ex)
        {
            Console.WriteLine(ex.Message);
            Console.WriteLine($"The queue {queueName} was not found.");
        }
    }
}
```

- For API details, see [GetQueueUrl](#) in [AWS SDK for .NET API Reference](#).

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get the URL for an Amazon SQS queue.

```
package main

import (
    "context"
    "flag"
    "fmt"
```

```
"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/sqs"
)

// SQSGetQueueUrlAPI defines the interface for the GetQueueUrl function.
// We use this interface to test the function using a mocked service.
type SQSGetQueueUrlAPI interface {
    GetQueueUrl(ctx context.Context,
        params *sqs.GetQueueUrlInput,
        optFns ...func(*sqs.Options)) (*sqs.GetQueueUrlOutput, error)
}

// GetQueueURL gets the URL of an Amazon SQS queue.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If success, a GetQueueUrlOutput object containing the result of the service
//     call and nil.
//     Otherwise, nil and an error from the call to GetQueueUrl.
func GetQueueURL(c context.Context, api SQSGetQueueUrlAPI, input
    *sqs.GetQueueUrlInput) (*sqs.GetQueueUrlOutput, error) {
    return api.GetQueueUrl(c, input)
}

func main() {
    queue := flag.String("q", "", "The name of the queue")
    flag.Parse()

    if *queue == "" {
        fmt.Println("You must supply a queue name (-q QUEUE")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := sqs.NewFromConfig(cfg)

    input := &sqs.GetQueueUrlInput{
        QueueName: queue,
    }

    result, err := GetQueueURL(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error getting the queue URL:")
        fmt.Println(err)
        return
    }

    fmt.Println("URL: " + *result.QueueUrl)
}
```

- For API details, see [GetQueueUrl](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
        GetQueueUrlResponse getQueueUrlResponse =
    sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        return getQueueUrlResponse.queueUrl();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [GetQueueUrl](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Get the URL for an Amazon SQS queue.

```
// Import required AWS SDK clients and commands for Node.js
import { GetQueueUrlCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = { QueueName: "SQS_QUEUE_NAME" };

const run = async () => {
    try {
        const data = await sqsClient.send(new GetQueueUrlCommand(params));
        console.log("Success", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetQueueUrl](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get the URL for an Amazon SQS queue.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {
  QueueName: 'SQS_QUEUE_NAME'
};

sqs.getQueueUrl(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetQueueUrl](#) in [AWS SDK for JavaScript API Reference](#).

## Python

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_queue(name):
    """
    Gets an SQS queue by name.

    :param name: The name that was used to create the queue.
    :return: A Queue object.
    """
    try:
        queue = sqs.get_queue_by_name(QueueName=name)
        logger.info("Got queue '%s' with URL=%s", name, queue.url)
    except ClientError as error:
        logger.exception("Couldn't get queue named %s.", name)
        raise error
    else:
        return queue
```

- For API details, see [GetQueueUrl](#) in [AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

## SDK for SAP ABAP

### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sqs->getqueueurl( iv_queuename = iv_queue_name ).      "  
oo_result is returned for testing purpose"  
    MESSAGE 'Queue URL retrieved' TYPE 'I'.  
CATCH /aws1/cx_sqqueuedoesnotexist.  
    MESSAGE 'The requested queue does not exist' TYPE 'E'.  
ENDTRY.
```

- For API details, see [GetQueueUrl](#) in *AWS SDK for SAP ABAP API reference*.

## List Amazon SQS queues using an AWS SDK

The following code examples show how to list Amazon SQS queues.

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List your Amazon SQS queues.

```
package main  
  
import (  
    "context"  
    "fmt"  
  
    "github.com/aws/aws-sdk-go-v2/config"  
    "github.com/aws/aws-sdk-go-v2/service/sqs"  
)  
  
// SQSListQueuesAPI defines the interface for the ListQueues function.  
// We use this interface to test the function using a mocked service.  
type SQSListQueuesAPI interface {  
    ListQueues(ctx context.Context,  
        params *sqs.ListQueuesInput,  
        optFns ...func(*sqs.Options)) (*sqs.ListQueuesOutput, error)  
}  
  
// GetQueues retrieves a list of your Amazon Simple Queue Service (Amazon SQS)  
// queues.  
// Inputs:  
//     c is the context of the method call, which includes the AWS Region.  
//     api is the interface that defines the method call.
```

```
//      input defines the input arguments to the service call.
// Output:
//      If success, a ListQueuesOutput object containing the result of the service
//      call and nil.
//      Otherwise, nil and an error from the call to ListQueues.
func GetQueues(c context.Context, api SQSListQueuesAPI, input *sqrs.ListQueuesInput)
(*sqrs.ListQueuesOutput, error) {
    return api.ListQueues(c, input)
}

func main() {
    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := sqs.NewFromConfig(cfg)

    input := &sqrs.ListQueuesInput{}

    result, err := GetQueues(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error retrieving queue URLs:")
        fmt.Println(err)
        return
    }

    for i, url := range result.QueueUrls {
        fmt.Printf("%d: %s\n", i+1, url)
    }
}
```

- For API details, see [ListQueues in AWS SDK for Go API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqscClient.listQueues(listQueuesRequest);
    for (String url : listQueuesResponse.queueUrls()) {
        System.out.println(url);
    }
} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

- For API details, see [ListQueues in AWS SDK for Java 2.x API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

List your Amazon SQS queues.

```
// Import required AWS SDK clients and commands for Node.js
import { ListQueuesCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

const run = async () => {
  try {
    const data = await sqsClient.send(new ListQueuesCommand({}));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.error(err, err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListQueues in AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List your Amazon SQS queues.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {};

sqs.listQueues(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrls);
  }
});
```

```
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListQueues in AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listQueues() {
    println("\nList Queues")

    val prefix = "que"
    val listQueuesRequest = ListQueuesRequest {
        queueNamePrefix = prefix
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.listQueues(listQueuesRequest)
        response.queueUrls?.forEach { url ->
            println(url)
        }
    }
}
```

- For API details, see [ListQueues in AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_queues(prefix=None):
    """
    Gets a list of SQS queues. When a prefix is specified, only queues with names
    that start with the prefix are returned.

    :param prefix: The prefix used to restrict the list of returned queues.
    :return: A list of Queue objects.
    """
    if prefix:
        queue_iter = sqs.queues.filter(QueueNamePrefix=prefix)
    else:
        queue_iter = sqs.queues.all()
    queues = list(queue_iter)
    if queues:
        logger.info("Got queues: %s", ', '.join([q.url for q in queues]))
```

```
else:  
    logger.warning("No queues found.")  
return queues
```

- For API details, see [ListQueues in AWS SDK for Python \(Boto3\) API Reference](#).

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Retrieve the first Amazon SQS queue listed in the Region.

```
async fn find_first_queue(client: &Client) -> Result<String, Error> {  
    let queues = client.list_queues().send().await?;  
    let queue_urls = queues.queue_urls().unwrap_or_default();  
    Ok(queue_urls  
        .first()  
        .expect("No queues in this account and Region. Create a queue to proceed.")  
        .to_string())  
}
```

- For API details, see [ListQueues in AWS SDK for Rust API reference](#).

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sqe->listqueues( ).           " oo_result is returned for  
testing purpose "  
    MESSAGE 'Retrieved list of queue(s)' TYPE 'I'.  
ENDTRY.
```

- For API details, see [ListQueues in AWS SDK for SAP ABAP API reference](#).

## Receive messages from an Amazon SQS queue using an AWS SDK

The following code examples show how to receive messages from an Amazon SQS queue.

.NET

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SQS;
using Amazon.SQS.Model;

public class ReceiveFromQueue
{
    public static async Task Main(string[] args)
    {
        string queueUrl = "https://sqs.us-east-2.amazonaws.com/0123456789ab/
Example_Queue";
        var attributeNames = new List<string>() { "All" };
        int maxNumberOfMessages = 5;
        var visibilityTimeout = (int)TimeSpan.FromMinutes(10).TotalSeconds;
        var waitTimeSeconds = (int)TimeSpan.FromSeconds(5).TotalSeconds;

        // If the Amazon SQS message queue is not in the same AWS Region as
your
        // default user, you need to provide the AWS Region as a parameter to
the
        // client constructor.
        var client = new AmazonSQSClient();

        var request = new ReceiveMessageRequest
        {
            QueueUrl = queueUrl,
            AttributeNames = attributeNames,
            MaxNumberOfMessages = maxNumberOfMessages,
            VisibilityTimeout = visibilityTimeout,
            WaitTimeSeconds = waitTimeSeconds,
        };

        var response = await client.ReceiveMessageAsync(request);

        if (response.Messages.Count > 0)
        {
            DisplayMessages(response.Messages);
        }
    }

    /// <summary>
    /// Display message information for a list of Amazon SQS messages.
    /// </summary>
    /// <param name="messages">The list of Amazon SQS Message objects to
display.</param>
    public static void DisplayMessages(List<Message> messages)
    {
        messages.ForEach(m =>
        {
            Console.WriteLine($"For message ID {m.MessageId}:");
            Console.WriteLine($" Body: {m.Body}");
            Console.WriteLine($" Receipt handle: {m.ReceiptHandle}");
            Console.WriteLine($" MD5 of body: {m.MD5OfBody}");
            Console.WriteLine($" MD5 of message attributes:
{m.MD5OfMessageAttributes}");
        });
    }
}
```

```
        Console.WriteLine("  Attributes:");
        foreach (var attr in m.Attributes)
        {
            Console.WriteLine($"    \t {attr.Key}: {attr.Value}");
        }
    });
}
```

Receive a message from an Amazon SQS queue, and then delete the message.

```

public static async Task Main()
{
    // If the AWS Region you want to use is different from
    // the AWS Region defined for the default user, supply
    // the specify your AWS Region to the client constructor.
    var client = new AmazonSQSClient();
    string queueName = "Example_Queue";

    var queueUrl = await GetQueueUrl(client, queueName);
    Console.WriteLine($"The SQS queue's URL is {queueUrl}");

    var response = await ReceiveAndDeleteMessage(client, queueUrl);

    Console.WriteLine($"Message: {response.Messages[0]}");
}

/// <summary>
/// Retrieve the queue URL for the queue named in the queueName
/// property using the client object.
/// </summary>
/// <param name="client">The Amazon SQS client used to retrieve the
/// queue URL.</param>
/// <param name="queueName">A string representing name of the queue
/// for which to retrieve the URL.</param>
/// <returns>The URL of the queue.</returns>
public static async Task<string> GetQueueUrl(IAmazonSQS client, string
queueName)
{
    var request = new GetQueueUrlRequest
    {
        QueueName = queueName,
    };

    GetQueueUrlResponse response = await client.GetQueueUrlAsync(request);
    return response.QueueUrl;
}

/// <summary>
/// Retrieves the message from the queue at the URL passed in the
/// queueURL parameters using the client.
/// </summary>
/// <param name="client">The SQS client used to retrieve a message.</param>
/// <param name="queueUrl">The URL of the queue from which to retrieve
/// a message.</param>
/// <returns>The response from the call to ReceiveMessageAsync.</returns>
public static async Task<ReceiveMessageResponse>
ReceiveAndDeleteMessage(IAmazonSQS client, string queueUrl)
{
    // Receive a single message from the queue.
    var receiveMessageRequest = new ReceiveMessageRequest
    {

```

```
        AttributeNames = { "SentTimestamp" },
        MaxNumberOfMessages = 1,
        MessageAttributeNames = { "All" },
        QueueUrl = queueUrl,
        VisibilityTimeout = 0,
        WaitTimeSeconds = 0,
    );
}

var receiveMessageResponse = await
client.ReceiveMessageAsync(receiveMessageRequest);

// Delete the received message from the queue.
var deleteMessageRequest = new DeleteMessageRequest
{
    QueueUrl = queueUrl,
    ReceiptHandle = receiveMessageResponse.Messages[0].ReceiptHandle,
};

await client.DeleteMessageAsync(deleteMessageRequest);

return receiveMessageResponse;
}
}
```

- For API details, see [ReceiveMessage](#) in [AWS SDK for .NET API Reference](#).

Go

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Receive a message from an Amazon SQS queue.

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SQSReceiveMessageAPI defines the interface for the GetQueueUrl function.
// We use this interface to test the function using a mocked service.
type SQSReceiveMessageAPI interface {
    GetQueueUrl(ctx context.Context,
        params *sqs.GetQueueUrlInput,
        optFns ...func(*sqs.Options)) (*sqs.GetQueueUrlOutput, error)

    ReceiveMessage(ctx context.Context,
        params *sqs.ReceiveMessageInput,
        optFns ...func(*sqs.Options)) (*sqs.ReceiveMessageOutput, error)
}

// GetQueueURL gets the URL of an Amazon SQS queue.
```

```
// Inputs:  
//   c is the context of the method call, which includes the AWS Region.  
//   api is the interface that defines the method call.  
//   input defines the input arguments to the service call.  
// Output:  
//   If success, a GetQueueUrlOutput object containing the result of the service  
//   call and nil.  
//   Otherwise, nil and an error from the call to GetQueueUrl.  
func GetQueueURL(c context.Context, api SQSReceiveMessageAPI, input  
  *sqrs.GetQueueUrlInput) (*sqrs.GetQueueUrlOutput, error) {  
    return api.GetQueueUrl(c, input)  
}  
  
// GetMessages gets the most recent message from an Amazon SQS queue.  
// Inputs:  
//   c is the context of the method call, which includes the AWS Region.  
//   api is the interface that defines the method call.  
//   input defines the input arguments to the service call.  
// Output:  
//   If success, a ReceiveMessageOutput object containing the result of the  
//   service call and nil.  
//   Otherwise, nil and an error from the call to ReceiveMessage.  
func GetMessages(c context.Context, api SQSReceiveMessageAPI, input  
  *sqrs.ReceiveMessageInput) (*sqrs.ReceiveMessageOutput, error) {  
    return api.ReceiveMessage(c, input)  
}  
  
func main() {  
    queue := flag.String("q", "", "The name of the queue")  
    timeout := flag.Int("t", 5, "How long, in seconds, that the message is hidden from  
    others")  
    flag.Parse()  
  
    if *queue == "" {  
        fmt.Println("You must supply the name of a queue (-q QUEUE)")  
        return  
    }  
  
    if *timeout < 0 {  
        *timeout = 0  
    }  
  
    if *timeout > 12*60*60 {  
        *timeout = 12 * 60 * 60  
    }  
  
    cfg, err := config.LoadDefaultConfig(context.TODO())  
    if err != nil {  
        panic("configuration error, " + err.Error())  
    }  
  
    client := sqs.NewFromConfig(cfg)  
  
    gQInput := &sqrs.GetQueueUrlInput{  
        QueueName: queue,  
    }  
  
    // Get URL of queue  
    urlResult, err := GetQueueURL(context.TODO(), client, gQInput)  
    if err != nil {  
        fmt.Println("Got an error getting the queue URL:")  
        fmt.Println(err)  
        return  
    }  
  
    queueURL := urlResult.QueueUrl
```

```
gMInput := &sqs.ReceiveMessageInput{
    MessageAttributeNames: []string{
        string(types.QueueAttributeNameAll),
    },
    QueueUrl:           queueURL,
    MaxNumberOfMessages: 1,
    VisibilityTimeout:   int32(*timeout),
}

msgResult, err := GetMessages(context.TODO(), client, gMInput)
if err != nil {
    fmt.Println("Got an error receiving messages:")
    fmt.Println(err)
    return
}

if msgResult.Messages != nil {
    fmt.Println("Message ID: " + *msgResult.Messages[0].MessageId)
    fmt.Println("Message Handle: " + *msgResult.Messages[0].ReceiptHandle)
} else {
    fmt.Println("No messages found")
}
}
```

- For API details, see [ReceiveMessage](#) in *AWS SDK for Go API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
    .queueUrl(queueUrl)
    .maxNumberOfMessages(5)
    .build();
return sqsClient.receiveMessage(receiveMessageRequest).messages();

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
```

- For API details, see [ReceiveMessage](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Receive a message from an Amazon SQS queue using long-poll support.

```
// Import required AWS SDK clients and commands for Node.js
import { ReceiveMessageCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const queueURL = "SQS_QUEUE_URL"; // SQS_QUEUE_URL
const params = {
    AttributeNames: ["SentTimestamp"],
    MaxNumberOfMessages: 1,
    MessageAttributeNames: ["All"],
    QueueUrl: queueURL,
    WaitTimeSeconds: 20,
};

const run = async () => {
    try {
        const data = await sqsClient.send(new ReceiveMessageCommand(params));
        console.log("Success, ", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ReceiveMessage in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Receive a message from an Amazon SQS queue using long-poll support.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var queueURL = "SQS_QUEUE_URL";

var params = {
    AttributeNames: [
        "SentTimestamp"
```

```
        ],
        MaxNumberOfMessages: 1,
        MessageAttributeNames: [
            "All"
        ],
        QueueUrl: queueURL,
        WaitTimeSeconds: 20
    };

    sqs.receiveMessage(params, function(err, data) {
        if (err) {
            console.log("Error", err);
        } else {
            console.log("Success", data);
        }
    });
}
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ReceiveMessage](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun receiveMessages(queueUrlVal: String?) {

    println("Retrieving messages from $queueUrlVal")

    val receiveMessageRequest = ReceiveMessageRequest {
        queueUrl = queueUrlVal
        maxNumberOfMessages = 5
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.receiveMessage(receiveMessageRequest)
        response.messages?.forEach { message ->
            println(message.body)
        }
    }
}
```

- For API details, see [ReceiveMessage](#) in [AWS SDK for Kotlin API reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def receive_messages(queue, max_number, wait_time):
    """
    Receive a batch of messages in a single request from an SQS queue.

    :param queue: The queue from which to receive messages.
    :param max_number: The maximum number of messages to receive. The actual number
                       of messages received might be less.
    :param wait_time: The maximum time to wait (in seconds) before returning. When
                      this number is greater than zero, long polling is used. This
                      can result in reduced costs and fewer false empty responses.
    :return: The list of Message objects received. These each contain the body
             of the message and metadata and custom attributes.
    """
    try:
        messages = queue.receive_messages(
            MessageAttributeNames=['All'],
            MaxNumberOfMessages=max_number,
            WaitTimeSeconds=wait_time
        )
        for msg in messages:
            logger.info("Received message: %s: %s", msg.message_id, msg.body)
    except ClientError as error:
        logger.exception("Couldn't receive messages from queue: %s", queue)
        raise error
    else:
        return messages
```

- For API details, see [ReceiveMessage](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn receive(client: &Client, queue_url: &String) -> Result<(), Error> {
    let rcv_message_output =
        client.receive_message().queue_url(queue_url).send().await?;

    println!("Messages from queue with url: {}", queue_url);

    for message in rcv_message_output.messages.unwrap_or_default() {
        println!("Got the message: {:#?}", message);
    }

    Ok(())
}
```

- For API details, see [ReceiveMessage](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Receive a message from an Amazon SQS queue.

```
TRY.  
    oo_result = lo_sqs->receivemessage( iv_queueurl = iv_queue_url ).      "  
oo_result is returned for testing purpose "  
    DATA(lt_messages) = oo_result->get_messages( ).  
    MESSAGE 'Message received from SQS queue' TYPE 'I'.  
    CATCH /aws1/cx_sqsoverlimit.  
    MESSAGE 'Maximum number of in flight messages reached' TYPE 'E'.  
ENDTRY.
```

Receive a message from an Amazon SQS queue using long-poll support.

```
TRY.  
    oo_result = lo_sqs->receivemessage(           " oo_result is returned for  
testing purpose "  
        iv_queueurl = iv_queue_url  
        iv_waittimeseconds = iv_wait_time      " time in seconds for long  
polling i.e. time for which the call waits for a message to arrive in queue before  
returning "  
    ).  
    DATA(lt_messages) = oo_result->get_messages( ).  
    MESSAGE 'Message received from SQS queue' TYPE 'I'.  
    CATCH /aws1/cx_sqsoverlimit.  
    MESSAGE 'Maximum number of in flight messages reached' TYPE 'E'.  
ENDTRY.
```

- For API details, see [ReceiveMessage](#) in *AWS SDK for SAP ABAP API reference*.

## Send a batch of messages to an Amazon SQS queue using an AWS SDK

The following code examples show how to send a batch of messages to an Amazon SQS queue.

### Java

#### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
SendMessageBatchRequest sendMessageBatchRequest =  
SendMessageBatchRequest.builder()  
    .queueUrl(queueUrl)
```

```
.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg 2").delaySeconds(10).build())
    .build();
sqscClient.sendMessageBatch(sendMessageBatchRequest);
```

- For API details, see [SendMessageBatch](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def send_messages(queue, messages):
    """
    Send a batch of messages in a single request to an SQS queue.
    This request may return overall success even when some messages were not sent.
    The caller must inspect the Successful and Failed lists in the response and
    resend any failed messages.

    :param queue: The queue to receive the messages.
    :param messages: The messages to send to the queue. These are simplified to
                     contain only the message body and attributes.
    :return: The response from SQS that contains the list of successful and failed
            messages.
    """
    try:
        entries = [
            {
                'Id': str(ind),
                'MessageBody': msg['body'],
                'MessageAttributes': msg['attributes']
            } for ind, msg in enumerate(messages)]
        response = queue.send_messages(Entries=entries)
        if 'Successful' in response:
            for msg_meta in response['Successful']:
                logger.info(
                    "Message sent: %s: %s",
                    msg_meta['MessageId'],
                    messages[int(msg_meta['Id'])]['body']
                )
        if 'Failed' in response:
            for msg_meta in response['Failed']:
                logger.warning(
                    "Failed to send: %s: %s",
                    msg_meta['MessageId'],
                    messages[int(msg_meta['Id'])]['body']
                )
    except ClientError as error:
        logger.exception("Send messages failed to queue: %s", queue)
        raise error
    else:
        return response
```

- For API details, see [SendMessageBatch](#) in *AWS SDK for Python (Boto3) API Reference*.

## Send a message to an Amazon SQS queue using an AWS SDK

The following code examples show how to send a message to an Amazon SQS queue.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.SQS;
using Amazon.SQS.Model;

public class SendMessageToQueue
{
    /// <summary>
    /// Initialize the Amazon SQS client object and use the
    /// SendMessageAsync method to send a message to an Amazon SQS queue.
    /// </summary>
    public static async Task Main()
    {
        string messageBody = "This is a sample message to send to the example
queue.";
        string queueUrl = "https://sqs.us-east-2.amazonaws.com/0123456789ab/
Example_Queue";

        // Create an Amazon SQS client object using the
        // default user. If the AWS Region you want to use
        // is different, supply the AWS Region as a parameter.
        IAmazonSQS client = new AmazonSQSClient();

        var request = new SendMessageRequest
        {
            MessageBody = messageBody,
            QueueUrl = queueUrl,
        };

        var response = await client.SendMessageAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully sent message. Message ID:
{response.MessageId}");
        }
        else
        {
            Console.WriteLine("Could not send message.");
        }
    }
}
```

Create an Amazon SQS queue and send a message to it.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
```

```

using Amazon;
using Amazon.SQS;
using Amazon.SQS.Model;

public class CreateSendExample
{
    // Specify your AWS Region (an example Region is shown).
    private static readonly string QueueName = "Example_Queue";
    private static readonly RegionEndpoint ServiceRegion =
RegionEndpoint.USWest2;
    private static IAmazonSQS client;

    public static async Task Main()
    {
        client = new AmazonSQSClient(ServiceRegion);
        var createQueueResponse = await CreateQueue(client, QueueName);

        string queueUrl = createQueueResponse.QueueUrl;

        Dictionary<string, MessageAttributeValue> messageAttributes = new
Dictionary<string, MessageAttributeValue>
        {
            { "Title", new MessageAttributeValue { DataType = "String",
StringValue = "The Whistler" } },
            { "Author", new MessageAttributeValue { DataType = "String",
StringValue = "John Grisham" } },
            { "WeeksOn", new MessageAttributeValue { DataType = "Number",
StringValue = "6" } };
        };

        string messageBody = "Information about current NY Times fiction
bestseller for week of 12/11/2016.";

        var sendMsgResponse = await SendMessage(client, queueUrl, messageBody,
messageAttributes);
    }

    ///<summary>
    /// Creates a new Amazon SQS queue using the queue name passed to it
    /// in queueName.
    ///</summary>
    ///<param name="client">An SQS client object used to send the message.</
param>
    ///<param name="queueName">A string representing the name of the queue
    /// to create.</param>
    ///<returns>A CreateQueueResponse that contains information about the
    /// newly created queue.</returns>
    public static async Task<CreateQueueResponse> CreateQueue(IAmazonSQS
client, string queueName)
    {
        var request = new CreateQueueRequest
        {
            QueueName = queueName,
            Attributes = new Dictionary<string, string>
            {
                { "DelaySeconds", "60" },
                { "MessageRetentionPeriod", "86400" },
            },
        };

        var response = await client.CreateQueueAsync(request);
        Console.WriteLine($"Created a queue with URL : {response.QueueUrl}");

        return response;
    }
}

```

```
    ///<summary>
    ///<summary>Sends a message to an SQS queue.
    ///</summary>
    ///<param name="client">An SQS client object used to send the message.</param>
    ///<param name="queueUrl">The URL of the queue to which to send the message.</param>
    ///<param name="messageBody">A string representing the body of the message to be sent to the queue.</param>
    ///<param name="messageAttributes">Attributes for the message to be sent to the queue.</param>
    ///<returns>A SendMessageResponse object that contains information about the message that was sent.</returns>
    public static async Task<SendMessageResponse> SendMessage(
        IAmazonSQS client,
        string queueUrl,
        string messageBody,
        Dictionary<string, MessageAttributeValue> messageAttributes)
    {
        var sendMessageRequest = new SendMessageRequest
        {
            DelaySeconds = 10,
            MessageAttributes = messageAttributes,
            MessageBody = messageBody,
            QueueUrl = queueUrl,
        };

        var response = await client.SendMessageAsync(sendMessageRequest);
        Console.WriteLine($"Sent a message with id : {response.MessageId}");

        return response;
    }
}
```

- For API details, see [SendMessage](#) in *AWS SDK for .NET API Reference*.

## Go

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Send a message to an Amazon SQS queue.

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SQSSendMessageAPI defines the interface for the GetQueueUrl and SendMessage functions.
// We use this interface to test the functions using a mocked service.
```

```
type SQSSendMessageAPI interface {
    GetQueueUrl(ctx context.Context,
        params *sns.GetQueueUrlInput,
        optFns ...func(*sns.Options)) (*sns.GetQueueUrlOutput, error)

    SendMessage(ctx context.Context,
        params *sns.SendMessageInput,
        optFns ...func(*sns.Options)) (*sns.SendMessageOutput, error)
}

// GetQueueURL gets the URL of an Amazon SQS queue.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If success, a GetQueueUrlOutput object containing the result of the service
//   call and nil.
//   Otherwise, nil and an error from the call to GetQueueUrl.
func GetQueueURL(c context.Context, api SQSSendMessageAPI, input
    *sns.GetQueueUrlInput) (*sns.GetQueueUrlOutput, error) {
    return api.GetQueueUrl(c, input)
}

// SendMsg sends a message to an Amazon SQS queue.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If success, a SendMessageOutput object containing the result of the service
//   call and nil.
//   Otherwise, nil and an error from the call to SendMessage.
func SendMsg(c context.Context, api SQSSendMessageAPI, input *sns.SendMessageInput)
    (*sns.SendMessageOutput, error) {
    return api.SendMessage(c, input)
}

func main() {
    queue := flag.String("q", "", "The name of the queue")
    flag.Parse()

    if *queue == "" {
        fmt.Println("You must supply the name of a queue (-q QUEUE)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := sns.NewFromConfig(cfg)

    // Get URL of queue
    gQInput := &sns.GetQueueUrlInput{
        QueueName: queue,
    }

    result, err := GetQueueURL(context.TODO(), client, gQInput)
    if err != nil {
        fmt.Println("Got an error getting the queue URL:")
        fmt.Println(err)
        return
    }
}
```

```
queueURL := result.QueueUrl

sMInput := &sqs.SendMessageInput{
    DelaySeconds: 10,
    MessageAttributes: map[string]types.MessageAttributeValue{
        "Title": {
            DataType: aws.String("String"),
            StringValue: aws.String("The Whistler"),
        },
        "Author": {
            DataType: aws.String("String"),
            StringValue: aws.String("John Grisham"),
        },
        "WeeksOn": {
            DataType: aws.String("Number"),
            StringValue: aws.String("6"),
        },
    },
    MessageBody: aws.String("Information about the NY Times fiction bestseller for
the week of 12/11/2016."),
    QueueUrl: queueURL,
}

resp, err := SendMsg(context.TODO(), client, sMInput)
if err != nil {
    fmt.Println("Got an error sending the message:")
    fmt.Println(err)
    return
}

fmt.Println("Sent message with ID: " + *resp.MessageId)
}
```

- For API details, see [SendMessage](#) in [AWS SDK for Go API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public class SendReceiveMessages {
    private static final String QUEUE_NAME = "testQueue" + new Date().getTime();

    public static void main(String[] args) {

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
            .build();

        try {
            CreateQueueRequest request = CreateQueueRequest.builder()
                .queueName(QUEUE_NAME)
                .build();
            CreateQueueResponse createResult = sqsClient.createQueue(request);

            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(QUEUE_NAME)
                .build();
        }
    }
}
```

```
String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();

SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
    .queueUrl(queueUrl)
    .messageBody("hello world")
    .delaySeconds(5)
    .build();
sqsClient.sendMessage(sendMsgRequest);

// Send multiple messages to the queue
SendMessageBatchRequest sendBatchRequest =
    SendMessageBatchRequest.builder()
        .queueUrl(queueUrl)
        .entries(
            SendMessageBatchRequestEntry.builder()
                .messageBody("Hello from message 1")
                .id("msg_1")
                .build(),
            SendMessageBatchRequestEntry.builder()
                .messageBody("Hello from message 2")
                .delaySeconds(10)
                .id("msg_2")
                .build())
        .build();
sqsClient.sendMessageBatch(sendBatchRequest);

// Receive messages from the queue
ReceiveMessageRequest receiveRequest = ReceiveMessageRequest.builder()
    .queueUrl(queueUrl)
    .build();
List<Message> messages =
    sqsClient.receiveMessage(receiveRequest).messages();

// Print out the messages
for (Message m : messages) {
    System.out.println("\n" +m.body());
}
} catch (QueueNameExistsException e) {
    throw e;
}
}
```

- For API details, see [SendMessage](#) in *AWS SDK for Java 2.x API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Send a message to an Amazon SQS queue.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessageCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = {
  DelaySeconds: 10,
  MessageAttributes: {
    Title: {
      DataType: "String",
      StringValue: "The Whistler",
    },
    Author: {
      DataType: "String",
      StringValue: "John Grisham",
    },
    WeeksOn: {
      DataType: "Number",
      StringValue: "6",
    },
  },
  MessageBody:
    "Information about current NY Times fiction bestseller for week of
12/11/2016.",
  // MessageDeduplicationId: "TheWhistler", // Required for FIFO queues
  // MessageGroupId: "Group1", // Required for FIFO queues
  QueueUrl: "SQS_QUEUE_URL" //SQS_QUEUE_URL; e.g., 'https://
sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'
};

const run = async () => {
  try {
    const data = await sqsClient.send(new SendMessageCommand(params));
    console.log("Success, message sent. MessageID:", data.MessageId);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SendMessage in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Send a message to an Amazon SQS queue.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});
```

```
var params = {
    // Remove DelaySeconds parameter and value for FIFO queues
    DelaySeconds: 10,
    MessageAttributes: {
        "Title": {
            DataType: "String",
            StringValue: "The Whistler"
        },
        "Author": {
            DataType: "String",
            StringValue: "John Grisham"
        },
        "WeeksOn": {
            DataType: "Number",
            StringValue: "6"
        }
    },
    MessageBody: "Information about current NY Times fiction bestseller for week of
12/11/2016.",
    // MessageDeduplicationId: "TheWhistler", // Required for FIFO queues
    // MessageGroupId: "Group1", // Required for FIFO queues
    QueueUrl: "SQS_QUEUE_URL"
};

sqS.sendMessage(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data.MessageId);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SendMessage](#) in [AWS SDK for JavaScript API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun sendMessages(queueUrlVal: String, message: String) {
    println("Sending multiple messages")
    println("\nSend message")
    val sendRequest = SendMessageRequest {
        queueUrl = queueUrlVal
        messageBody = message
        delaySeconds = 10
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessage(sendRequest)
        println("A single message was successfully sent.")
    }
}
```

```
}

suspend fun sendBatchMessages(queueUrlVal: String?) {
    println("Sending multiple messages")

    val msg1 = SendMessageBatchRequestEntry {
        id = "id1"
        messageBody = "Hello from msg 1"
    }

    val msg2 = SendMessageBatchRequestEntry {
        id = "id2"
        messageBody = "Hello from msg 2"
    }

    val sendMessageBatchRequest = SendMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = listOf(msg1, msg2)
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessageBatch(sendMessageBatchRequest)
        println("Batch message were successfully sent.")
    }
}
```

- For API details, see [SendMessage](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def send_message(queue, message_body, message_attributes=None):
    """
    Send a message to an Amazon SQS queue.

    :param queue: The queue that receives the message.
    :param message_body: The body text of the message.
    :param message_attributes: Custom attributes of the message. These are key-
value
                           pairs that can be whatever you want.
    :return: The response from SQS that contains the assigned message ID.
    """
    if not message_attributes:
        message_attributes = {}

    try:
        response = queue.send_message(
            MessageBody=message_body,
            MessageAttributes=message_attributes
        )
    except ClientError as error:
        logger.exception("Send message failed: %s", message_body)
        raise error
    else:
        return response
```

- For API details, see [SendMessage](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn send(client: &Client, queue_url: &String, message: &SQSMessage) ->
Result<(), Error> {
    println!("Sending message to queue with URL: {}", queue_url);

    let rsp = client
        .send_message()
        .queue_url(queue_url)
        .message_body(&message.body)
        .message_group_id(&message.group)
        // If the queue is FIFO, you need to set .message_deduplication_id
        // or configure the queue for ContentBasedDeduplication.
        .send()
        .await?;

    println!("Send message to the queue: {:?}", rsp);
}

Ok(())
}
```

- For API details, see [SendMessage](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_sqc->sendmessage(                               " oo_result is returned for
testing purpose "
        iv_queueurl = iv_queue_url
        iv_messagebody = iv_message
    ).
    MESSAGE 'Message sent to SQS queue' TYPE 'I'.
CATCH /aws1/cx_sqsinvalidmsgconts.
    MESSAGE 'Message contains invalid characters' TYPE 'E'.
CATCH /aws1/cx_sqsunsupportedop.
    MESSAGE 'Operation not supported' TYPE 'E'.
```

ENDTRY.

- For API details, see [SendMessage](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios for Amazon SQS using AWS SDKs

The following code examples show how to use Amazon Simple Queue Service with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Send and receive batches of messages with Amazon SQS using an AWS SDK \(p. 2084\)](#)

## Send and receive batches of messages with Amazon SQS using an AWS SDK

The following code example shows how to:

- Create an Amazon SQS queue.
- Send batches of messages to the queue.
- Receive batches of messages from the queue.
- Delete batches of messages from the queue.

### Python

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions to wrap Amazon SQS message functions.

```
import logging
import sys

import boto3
from botocore.exceptions import ClientError

import queue_wrapper

logger = logging.getLogger(__name__)
sqS = boto3.resource('sqS')

def send_messages(queue, messages):
    """
        Send a batch of messages in a single request to an SQS queue.
        This request may return overall success even when some messages were not sent.
        The caller must inspect the Successful and Failed lists in the response and
        resend any failed messages.

        :param queue: The queue to receive the messages.
        :param messages: The messages to send to the queue. These are simplified to
                         contain only the message body and attributes.
        :return: The response from SQS that contains the list of successful and failed
    
```

```

        messages.

"""
try:
    entries = [
        'Id': str(ind),
        'MessageBody': msg['body'],
        'MessageAttributes': msg['attributes']
    } for ind, msg in enumerate(messages)]
    response = queue.send_messages(Entries=entries)
    if 'Successful' in response:
        for msg_meta in response['Successful']:
            logger.info(
                "Message sent: %s: %s",
                msg_meta['MessageId'],
                messages[int(msg_meta['Id'])]['body']
            )
    if 'Failed' in response:
        for msg_meta in response['Failed']:
            logger.warning(
                "Failed to send: %s: %s",
                msg_meta['MessageId'],
                messages[int(msg_meta['Id'])]['body']
            )
except ClientError as error:
    logger.exception("Send messages failed to queue: %s", queue)
    raise error
else:
    return response

def receive_messages(queue, max_number, wait_time):
    """
    Receive a batch of messages in a single request from an SQS queue.

    :param queue: The queue from which to receive messages.
    :param max_number: The maximum number of messages to receive. The actual number
                       of messages received might be less.
    :param wait_time: The maximum time to wait (in seconds) before returning. When
                      this number is greater than zero, long polling is used. This
                      can result in reduced costs and fewer false empty responses.
    :return: The list of Message objects received. These each contain the body
            of the message and metadata and custom attributes.
    """
    try:
        messages = queue.receive_messages(
            MessageAttributeNames=['All'],
            MaxNumberOfMessages=max_number,
            WaitTimeSeconds=wait_time
        )
        for msg in messages:
            logger.info("Received message: %s: %s", msg.message_id, msg.body)
    except ClientError as error:
        logger.exception("Couldn't receive messages from queue: %s", queue)
        raise error
    else:
        return messages

def delete_messages(queue, messages):
    """
    Delete a batch of messages from a queue in a single request.

    :param queue: The queue from which to delete the messages.
    :param messages: The list of messages to delete.
    :return: The response from SQS that contains the list of successful and failed
            message deletions.
    """
    try:

```

```

        entries = [
            {
                'Id': str(ind),
                'ReceiptHandle': msg.receipt_handle
            } for ind, msg in enumerate(messages)]
        response = queue.delete_messages(Entries=entries)
        if 'Successful' in response:
            for msg_meta in response['Successful']:
                logger.info("Deleted %s",
                            messages[int(msg_meta['Id'])].receipt_handle)
        if 'Failed' in response:
            for msg_meta in response['Failed']:
                logger.warning(
                    "Could not delete %s",
                    messages[int(msg_meta['Id'])].receipt_handle
                )
    except ClientError:
        logger.exception("Couldn't delete messages from queue %s", queue)
    else:
        return response

```

Use the wrapper functions to send and receive messages in batches.

```

def usage_demo():
    """
    Shows how to:
    * Read the lines from this Python file and send the lines in
      batches of 10 as messages to a queue.
    * Receive the messages in batches until the queue is empty.
    * Reassemble the lines of the file and verify they match the original file.
    """
    def pack_message(msg_path, msg_body, msg_line):
        return {
            'body': msg_body,
            'attributes': {
                'path': {'StringValue': msg_path, 'DataType': 'String'},
                'line': {'StringValue': str(msg_line), 'DataType': 'String'}
            }
        }

    def unpack_message(msg):
        return (msg.message_attributes['path']['StringValue'],
                msg.body,
                int(msg.message_attributes['line']['StringValue']))

    print('*'*88)
    print("Welcome to the Amazon Simple Queue Service (Amazon SQS) demo!")
    print('*'*88)

    queue = queue_wrapper.create_queue('sqss-usage-demo-message-wrapper')

    with open(__file__) as file:
        lines = file.readlines()

    line = 0
    batch_size = 10
    received_lines = [None]*len(lines)
    print(f"Sending file lines in batches of {batch_size} as messages.")
    while line < len(lines):
        messages = [pack_message(__file__, lines[index], index)
                   for index in range(line, min(line + batch_size, len(lines)))]
        line = line + batch_size
        send_messages(queue, messages)
        print('.')
        sys.stdout.flush()

```

```
print(f"Done. Sent {len(lines)} - 1} messages.")

print(f"Receiving, handling, and deleting messages in batches of
{batch_size}.")
more_messages = True
while more_messages:
    received_messages = receive_messages(queue, batch_size, 2)
    print('.', end='')
    sys.stdout.flush()
    for message in received_messages:
        path, body, line = unpack_message(message)
        received_lines[line] = body
    if received_messages:
        delete_messages(queue, received_messages)
    else:
        more_messages = False
print('Done.')

if all([lines[index] == received_lines[index] for index in range(len(lines))]):
    print(f"Successfully reassembled all file lines!")
else:
    print(f"Uh oh, some lines were missed!")

queue.delete()

print("Thanks for watching!")
print('*88)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateQueue](#)
  - [DeleteMessageBatch](#)
  - [DeleteQueue](#)
  - [ReceiveMessage](#)
  - [SendMessageBatch](#)

## Cross-service examples for Amazon SQS using AWS SDKs

The following code examples show how to use Amazon Simple Queue Service with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Create a web application that sends and retrieves messages by using the Amazon Simple Queue Service API \(p. 2087\)](#)
- [Create a messenger application with Step Functions \(p. 2088\)](#)
- [Create an Amazon Textract explorer application \(p. 2089\)](#)
- [Detect people and objects in a video with Amazon Rekognition using an AWS SDK \(p. 2090\)](#)

### Create a web application that sends and retrieves messages by using the Amazon Simple Queue Service API

The following code examples show how to create a messaging application by using the Amazon Simple Queue Service API.

Java

### SDK for Java 2.x

Shows how to use the Amazon Simple Queue Service API to develop a Spring REST API that sends and retrieves messages.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon SQS

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to use the Amazon Simple Queue Service API to develop a Spring REST API that sends and retrieves messages.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon SQS

## Create a messenger application with Step Functions

The following code example shows how to create an AWS Step Functions messenger application that retrieves message records from a database table.

Python

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with AWS Step Functions to create a messenger application that retrieves message records from an Amazon DynamoDB table and sends them with Amazon Simple Queue Service (Amazon SQS). The state machine integrates with an AWS Lambda function to scan the database for unsent messages.

- Create a state machine that retrieves and updates message records from an Amazon DynamoDB table.
- Update the state machine definition to also send messages to Amazon Simple Queue Service (Amazon SQS).
- Start and stop state machine runs.
- Connect to Lambda, DynamoDB, and Amazon SQS from a state machine by using service integrations.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Lambda
- Amazon SQS
- Step Functions

## Create an Amazon Textract explorer application

The following code examples show how to explore Amazon Textract output through an interactive application.

JavaScript

### SDK for JavaScript V3

Shows how to use the AWS SDK for JavaScript to build a React application that uses Amazon Textract to extract data from a document image and display it in an interactive web page. This example runs in a web browser and requires an authenticated Amazon Cognito identity for credentials. It uses Amazon Simple Storage Service (Amazon S3) for storage, and for notifications it polls an Amazon Simple Queue Service (Amazon SQS) queue that is subscribed to an Amazon Simple Notification Service (Amazon SNS) topic.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with Amazon Textract to detect text, form, and table elements in a document image. The input image and Amazon Textract output are shown in a Tkinter application that lets you explore the detected elements.

- Submit a document image to Amazon Textract and explore the output of detected elements.
- Submit images directly to Amazon Textract or through an Amazon Simple Storage Service (Amazon S3) bucket.
- Use asynchronous APIs to start a job that publishes a notification to an Amazon Simple Notification Service (Amazon SNS) topic when the job completes.
- Poll an Amazon Simple Queue Service (Amazon SQS) queue for a job completion message and display the results.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon S3

- Amazon SNS
- Amazon SQS
- Amazon Textract

## Detect people and objects in a video with Amazon Rekognition using an AWS SDK

The following code examples show how to detect people and objects in a video with Amazon Rekognition.

Python

### SDK for Python (Boto3)

Use Amazon Rekognition to detect faces, objects, and people in videos by starting asynchronous detection jobs. This example also configures Amazon Rekognition to notify an Amazon Simple Notification Service (Amazon SNS) topic when jobs complete and subscribes an Amazon Simple Queue Service (Amazon SQS) queue to the topic. When the queue receives a message about a job, the job is retrieved and the results are output.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon SNS
- Amazon SQS

## Code examples for AWS STS using AWS SDKs

The following code examples show how to use AWS Security Token Service with an AWS software development kit (SDK).

### Code examples

- [Actions for AWS STS using AWS SDKs \(p. 2090\)](#)
  - [Assume a role with AWS STS using an AWS SDK \(p. 2091\)](#)
  - [Get a session token with AWS STS using an AWS SDK \(p. 2096\)](#)
- [Scenarios for AWS STS using AWS SDKs \(p. 2097\)](#)
  - [Assume an IAM role that requires an MFA token with AWS STS using an AWS SDK \(p. 2097\)](#)
  - [Construct a URL with AWS STS for federated users using an AWS SDK \(p. 2101\)](#)
  - [Get a session token that requires an MFA token with AWS STS using an AWS SDK \(p. 2104\)](#)

## Actions for AWS STS using AWS SDKs

The following code examples show how to use AWS Security Token Service with AWS SDKs. Each example calls an individual service function.

### Examples

- [Assume a role with AWS STS using an AWS SDK \(p. 2091\)](#)

- Get a session token with AWS STS using an AWS SDK (p. 2096)

## Assume a role with AWS STS using an AWS SDK

The following code examples show how to assume a role with AWS STS.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Threading.Tasks;

namespace AssumeRoleExample
{
    class AssumeRole
    {
        ///<summary>
        /// This example shows how to use the AWS Security Token
        /// Service (AWS STS) to assume an IAM role.
        ///
        /// NOTE: It is important that the role that will be assumed has a
        /// trust relationship with the account that will assume the role.
        ///
        /// Before you run the example, you need to create the role you want to
        /// assume and have it trust the IAM account that will assume that role.
        ///
        /// See https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_roles\_create.html
        /// for help in working with roles.
        /// </summary>

        private static readonly RegionEndpoint REGION = RegionEndpoint.USWest2;

        static async Task Main()
        {
            // Create the SecurityToken client and then display the identity of the
            // default user.
            var roleArnToAssume = "arn:aws:iam::123456789012:role/testAssumeRole";

            var client = new
                Amazon.SecurityToken.AmazonSecurityTokenServiceClient(REGION);

            // Get and display the information about the identity of the default
            user.
            var callerIdRequest = new GetCallerIdentityRequest();
            var caller = await client.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"Original Caller: {caller.Arн}");

            // Create the request to use with the AssumeRoleAsync call.
            var assumeRoleReq = new AssumeRoleRequest()
            {
                DurationSeconds = 1600,
                RoleSessionName = "Session1",
                RoleArn = roleArnToAssume
            }
        }
}
```

```
};

        var assumeRoleRes = await client.AssumeRoleAsync(assumeRoleReq);

        // Now create a new client based on the credentials of the caller
        // assuming the role.
        var client2 = new AmazonSecurityTokenServiceClient(credentials:
assumeRoleRes.Credentials);

        // Get and display information about the caller that has assumed the
defined role.
        var caller2 = await client2.GetCallerIdentityAsync(callerIdRequest);
        Console.WriteLine($"AssumedRole Caller: {caller2.Arn}");
    }
}
```

- For API details, see [AssumeRole in AWS SDK for .NET API Reference](#).

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
                             const Aws::String &roleSessionName,
                             const Aws::String &externalId,
                             Aws::Auth::AWSCredentials &credentials,
                             const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::STS::STSClient sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
                    outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials =
result.GetCredentials();

        // Store temporary credentials in return argument.
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }

    return outcome.IsSuccess();
}
```

- For API details, see [AssumeRole in AWS SDK for C++ API Reference](#).

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon STS service client object.
const stsClient = new STSClient({ region: REGION });
export { stsClient };
```

Assume the IAM role.

```
// Import required AWS SDK clients and commands for Node.js
import { stsClient } from "./libs/stsClient.js";
import {
    AssumeRoleCommand,
    GetCallerIdentityCommand,
} from "@aws-sdk/client-sts";

// Set the parameters
export const params = {
    RoleArn: "ARN_OF_ROLE_TO_ASSUME", //ARN_OF_ROLE_TO_ASSUME
    RoleSessionName: "session1",
    DurationSeconds: 900,
};

export const run = async () => {
    try {
        //Assume Role
        const data = await stsClient.send(new AssumeRoleCommand(params));
        return data;
        const rolecreds = {
            accessKeyId: data.Credentials.AccessKeyId,
            secretAccessKey: data.Credentials.SecretAccessKey,
            sessionToken: data.Credentials.SessionToken,
        };
        //Get Amazon Resource Name (ARN) of current identity
        try {
            const stsParams = { credentials: rolecreds };
            const stsClient = new STSClient(stsParams);
            const results = await stsClient.send(
                new GetCallerIdentityCommand(rolecreds)
            );
            console.log("Success", results);
        } catch (err) {
            console.log(err, err.stack);
        }
    } catch (err) {
        console.log("Error", err);
    }
}
```

```
};  
run();
```

- For API details, see [AssumeRole](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
const AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
var roleToAssume = {RoleArn: 'arn:aws:iam::123456789012:role/RoleName',  
                    RoleSessionName: 'session1',  
                    DurationSeconds: 900,};  
var roleCreds;  
  
// Create the STS service object  
var sts = new AWS.STS({apiVersion: '2011-06-15'});  
  
//Assume Role  
sts.assumeRole(roleToAssume, function(err, data) {  
    if (err) console.log(err, err.stack);  
    else{  
        roleCreds = {accessKeyId: data.Credentials.AccessKeyId,  
                     secretAccessKey: data.Credentials.SecretAccessKey,  
                     sessionToken: data.Credentials.SessionToken};  
        stsGetCallerIdentity(roleCreds);  
    }  
});  
  
//Get Arn of current identity  
function stsGetCallerIdentity(creds) {  
    var stsParams = {credentials: creds };  
    // Create STS service object  
    var sts = new AWS.STS(stsParams);  
  
    sts.getCallerIdentity({}, function(err, data) {  
        if (err) {  
            console.log(err, err.stack);  
        }  
        else {  
            console.log(data.Arn);  
        }  
    });  
}
```

- For API details, see [AssumeRole](#) in *AWS SDK for JavaScript API Reference*.

## Python

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Assume an IAM role that requires an MFA token and use temporary credentials to list Amazon S3 buckets for the account.

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
    account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                           grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
                             MFA
                           device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    response = sts_clientassume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp)
    temp_credentials = response['Credentials']
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        's3',
        aws_access_key_id=temp_credentials['AccessKeyId'],
        aws_secret_access_key=temp_credentials['SecretAccessKey'],
        aws_session_token=temp_credentials['SessionToken'])

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- For API details, see [AssumeRole in AWS SDK for Python \(Boto3\) API Reference](#).

## Ruby

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates an AWS Security Token Service (AWS STS) client with specified
# credentials.
# This is separated into a factory function so that it can be mocked for unit
# testing.
#
# @param key_id [String] The ID of the access key used by the STS client.
# @param key_secret [String] The secret part of the access key used by the STS
# client.
def create_sts_client(key_id, key_secret)
  Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
end
```

```
# Gets temporary credentials that can be used to assume a role.  
#  
# @param role_arn [String] The ARN of the role that is assumed when these  
credentials  
# are used.  
# @param sts_client [AWS::STS::Client] An AWS STS client.  
# @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume  
the role.  
def assume_role(role_arn, sts_client)  
    credentials = Aws::AssumeRoleCredentials.new(  
        client: sts_client,  
        role_arn: role_arn,  
        role_session_name: "create-use-assume-role-scenario"  
    )  
    puts("Assumed role '#{role_arn}', got temporary credentials.")  
    credentials  
end
```

- For API details, see [AssumeRole in AWS SDK for Ruby API Reference](#).

## Get a session token with AWS STS using an AWS SDK

The following code example shows how to get a session token with AWS STS and use it to perform a service action that requires an MFA token.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get a session token by passing an MFA token and use it to list Amazon S3 buckets for the account.

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,  
sts_client):  
    """  
    Gets a session token with MFA credentials and uses the temporary session  
    credentials to list Amazon S3 buckets.  
  
    Requires an MFA device serial number and token.  
  
    :param mfa_serial_number: The serial number of the MFA device. For a virtual  
    MFA  
        device, this is an Amazon Resource Name (ARN).  
    :param mfa_totp: A time-based, one-time password issued by the MFA device.  
    :param sts_client: A Boto3 STS instance that has permission to assume the role.  
    """  
    if mfa_serial_number is not None:  
        response = sts_client.get_session_token(  
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp)  
    else:  
        response = sts_client.get_session_token()  
    temp_credentials = response['Credentials']  
  
    s3_resource = boto3.resource(  
        's3',  
        aws_access_key_id=temp_credentials['AccessKeyId'],
```

```
aws_secret_access_key=temp_credentials['SecretAccessKey'],
aws_session_token=temp_credentials['SessionToken'])

print(f"Buckets for the account:")
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

- For API details, see [GetSessionToken](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios for AWS STS using AWS SDKs

The following code examples show how to use AWS Security Token Service with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Assume an IAM role that requires an MFA token with AWS STS using an AWS SDK \(p. 2097\)](#)
- [Construct a URL with AWS STS for federated users using an AWS SDK \(p. 2101\)](#)
- [Get a session token that requires an MFA token with AWS STS using an AWS SDK \(p. 2104\)](#)

## Assume an IAM role that requires an MFA token with AWS STS using an AWS SDK

The following code example shows how to:

- Create an IAM role that grants permission to list Amazon S3 buckets.
- Create an IAM user that has permission to assume the role only when MFA credentials are provided.
- Register an MFA device for the user.
- Assume the role and use temporary credentials to list S3 buckets.

### Python

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an IAM user, register an MFA device, and create a role that grants permission to list S3 buckets. The user has rights only to assume the role.

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual MFA device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role and requires
    MFA.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

```

For demonstration purposes, the user is created in the same account as the role,  
but in practice the user would likely be from another account.

Any MFA device that can scan a QR code will work with this demonstration.  
Common choices are mobile apps like LastPass Authenticator,  
Microsoft Authenticator, or Google Authenticator.

```
:param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
                    that has permissions to create users, roles, and policies
                    in the account.
:return: The newly created user, user key, virtual MFA device, and role.
"""
user = iam_resource.create_user(UserName=unique_name('user'))
print(f"Created user {user.name}.")

virtual_mfa_device = iam_resource.create_virtual_mfa_device(
    VirtualMFADeviceName=unique_name('mfa'))
print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

print(f"Showing the QR code for the device. Scan this in the MFA app of your "
      f"choice.")
with open('qr.png', 'wb') as qr_file:
    qr_file.write(virtual_mfa_device.qr_code_png)
webbrowser.open(qr_file.name)

print(f"Enter two consecutive code from your MFA device.")
mfa_code_1 = input("Enter the first code: ")
mfa_code_2 = input("Enter the second code: ")
user.enable_mfa(
    SerialNumber=virtual_mfa_device.serial_number,
    AuthenticationCode1=mfa_code_1,
    AuthenticationCode2=mfa_code_2)
os.remove(qr_file.name)
print(f"MFA device is registered with the user.")

user_key = user.create_access_key_pair()
print(f"Created access key pair for user.")

print(f"Wait for user to be ready.", end='')
progress_bar(10)

role = iam_resource.create_role(
    RoleName=unique_name('role'),
    AssumeRolePolicyDocument=json.dumps({
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Principal': {'AWS': user.arn},
                'Action': 'sts:AssumeRole',
                'Condition': {'Bool': {'aws:MultiFactorAuthPresent': True}}
            }
        ]
    })
)
print(f"Created role {role.name} that requires MFA.")

policy = iam_resource.create_policy(
    PolicyName=unique_name('policy'),
    PolicyDocument=json.dumps({
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
```

```

        'Action': 's3>ListAllMyBuckets',
        'Resource': 'arn:aws:s3:::/*'
    }
]
})
role.attach_policy(PolicyArn=policy.arn)
print(f"Created policy {policy.policy_name} and attached it to the role.")

user.create_policy(
    PolicyName=unique_name('user-policy'),
    PolicyDocument=json.dumps({
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Action': 'sts:AssumeRole',
                'Resource': role.arn
            }
        ]
    })
)
print(f"Created an inline policy for {user.name} that lets the user assume "
      f"the role.")

print("Give AWS time to propagate these new resources and connections.",
      end='')
progress_bar(10)

return user, user_key, virtual_mfa_device, role

```

Show that assuming the role without an MFA token is not allowed.

```

def try_to_assume_role_without_mfa(assume_role_arn, session_name, sts_client):
    """
    Shows that attempting to assume the role without sending MFA credentials
    results
    in an AccessDenied error.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role to assume.
    :param session_name: The name of the STS session.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    print(f"Trying to assume the role without sending MFA credentials...")
    try:
        sts_client.assume_role(RoleArn=assume_role_arn,
                               RoleSessionName=session_name)
        raise RuntimeError("Expected AccessDenied error.")
    except ClientError as error:
        if error.response['Error']['Code'] == 'AccessDenied':
            print("Got AccessDenied.")
        else:
            raise

```

Assume the role that grants permission to list S3 buckets, passing the required MFA token, and show that buckets can be listed.

```

def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client):
    """
    Assumes a role from another account and uses the temporary credentials from

```

that role to list the Amazon S3 buckets that are owned by the other account.  
Requires an MFA device serial number and token.

The assumed role must grant permission to list the buckets in the other account.

```
:param assume_role_arn: The Amazon Resource Name (ARN) of the role that grants access to list the other account's buckets.
:param session_name: The name of the STS session.
:param mfa_serial_number: The serial number of the MFA device. For a virtual MFA
                           device, this is an ARN.
:param mfa_totp: A time-based, one-time password issued by the MFA device.
:param sts_client: A Boto3 STS instance that has permission to assume the role.
"""
response = sts_client.assume_role(
    RoleArn=assume_role_arn,
    RoleSessionName=session_name,
    SerialNumber=mfa_serial_number,
    TokenCode=mfa_totp)
temp_credentials = response['Credentials']
print(f"Assumed role {assume_role_arn} and got temporary credentials.")

s3_resource = boto3.resource(
    's3',
    aws_access_key_id=temp_credentials['AccessKeyId'],
    aws_secret_access_key=temp_credentials['SecretAccessKey'],
    aws_session_token=temp_credentials['SessionToken'])

print(f"Listing buckets for the assumed role's account:")
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

Destroy the resources created for the demo.

```
def teardown(user, virtual_mfa_device, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        policy_name = attached.policy_name
        role.detach_policy(PolicyArn=attached.arn)
        attached.delete()
        print(f"Detached and deleted {policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

Run this scenario by using the previously defined functions.

```
def usage_demo():
    """Drives the demonstration."""
    print('*'*88)
    print(f"Welcome to the AWS Security Token Service assume role demo, "
          f"starring multi-factor authentication (MFA)!")
    print('*'*88)
    iam_resource = boto3.resource('iam')
    user, user_key, virtual_mfa_device, role = setup(iam_resource)
    print(f"Created {user.name} and {role.name}.")
    try:
        sts_client = boto3.client(
            'sts', aws_access_key_id=user_key.id,
            aws_secret_access_key=user_key.secret)
        try_to_assume_role_without_mfa(role.arn, 'demo-sts-session', sts_client)
        mfa_totp = input('Enter the code from your registered MFA device: ')
        list_buckets_from_assumed_role_with_mfa(
            role.arn, 'demo-sts-session', virtual_mfa_device.serial_number,
            mfa_totp, sts_client)
    finally:
        teardown(user, virtual_mfa_device, role)
    print("Thanks for watching!")
```

- For API details, see [AssumeRole in AWS SDK for Python \(Boto3\) API Reference](#).

## Construct a URL with AWS STS for federated users using an AWS SDK

The following code example shows how to:

- Create an IAM role that grants read-only access to the current account's Amazon S3 resources.
- Get a security token from the AWS federation endpoint.
- Construct a URL that can be used to access the console with federated credentials.

### Python

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a role that grants read-only access to the current account's S3 resources.

```
def setup(iam_resource):
    """
    Creates a role that can be assumed by the current user.
    Attaches a policy that allows only Amazon S3 read-only access.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) instance
                         that has the permission to create a role.
    :return: The newly created role.
    """
    role = iam_resource.create_role(
        RoleName=unique_name('role'),
        AssumeRolePolicyDocument=json.dumps({
            'Version': '2012-10-17',
            'Statement': [
                {
                    'Effect': 'Allow',
                    'Action': [
                        's3:ListBucket',
                        's3:GetObject'
                    ],
                    'Resource': [
                        'arn:aws:s3:::' + unique_name('bucket'),
                        'arn:aws:s3:::' + unique_name('bucket') + '/*'
                    ]
                }
            ]
        })
    return role
```

```

        'Effect': 'Allow',
        'Principal': {'AWS': iam_resource.CurrentUser().arn},
        'Action': 'sts:AssumeRole'
    }
]
})
role.attach_policy(PolicyArn='arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess')
print(f"Created role {role.name}.")

print("Give AWS time to propagate these new resources and connections.",
end='')
progress_bar(10)

return role

```

Get a security token from the AWS federation endpoint and construct a URL that can be used to access the console with federated credentials.

```

def construct_federated_url(assume_role_arn, session_name, issuer, sts_client):
    """
    Constructs a URL that gives federated users direct access to the AWS Management Console.

    1. Acquires temporary credentials from AWS Security Token Service (AWS STS) that can be used to assume a role with limited permissions.
    2. Uses the temporary credentials to request a sign-in token from the AWS federation endpoint.
    3. Builds a URL that can be used in a browser to navigate to the AWS federation endpoint, includes the sign-in token for authentication, and redirects to the AWS Management Console with permissions defined by the role that was specified in step 1.

    :param assume_role_arn: The role that specifies the permissions that are granted.
                           The current user must have permission to assume the role.
    :param session_name: The name for the STS session.
    :param issuer: The organization that issues the URL.
    :param sts_client: A Boto3 STS instance that can assume the role.
    :return: The federated URL.
    """

    response = sts_client.assume_role(
        RoleArn=assume_role_arn, RoleSessionName=session_name)
    temp_credentials = response['Credentials']
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    session_data = {
        'sessionId': temp_credentials['AccessKeyId'],
        'sessionKey': temp_credentials['SecretAccessKey'],
        'sessionToken': temp_credentials['SessionToken']
    }
    aws_federated_signin_endpoint = 'https://signin.aws.amazon.com/federation'

    # Make a request to the AWS federation endpoint to get a sign-in token.
    # The requests.get function URL-encodes the parameters and builds the query string
    # before making the request.
    response = requests.get(
        aws_federated_signin_endpoint,
        params={
            'Action': 'getSigninToken',

```

```
'SessionDuration': str(datetime.timedelta(hours=12).seconds),
    'Session': json.dumps(session_data)
})
signin_token = json.loads(response.text)
print(f"Got a sign-in token from the AWS sign-in federation endpoint.")

# Make a federated URL that can be used to sign into the AWS Management
Console.
query_string = urllib.parse.urlencode({
    'Action': 'login',
    'Issuer': issuer,
    'Destination': 'https://console.aws.amazon.com/',
    'SigninToken': signin_token['SigninToken']
})
federated_url = f'{aws_federated_signin_endpoint}?{query_string}'
return federated_url
```

Destroy the resources created for the demo.

```
def teardown(role):
    """
    Removes all resources created during setup.

    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        role.detach_policy(PolicyArn=attached.arn)
        print(f"Detached {attached.policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
```

Run this scenario by using the previously defined functions.

```
def usage_demo():
    """Drives the demonstration."""
    print('*'*88)
    print(f"Welcome to the AWS Security Token Service federated URL demo.")
    print('*'*88)
    iam_resource = boto3.resource('iam')
    role = setup(iam_resource)
    sts_client = boto3.client('sts')
    try:
        federated_url = construct_federated_url(
            role.arn, 'AssumeRoleDemoSession', 'example.org', sts_client)
        print("Constructed a federated URL that can be used to connect to the "
              "AWS Management Console with role-defined permissions:")
        print(federated_url)
        print('*'*88)
        _ = input("Copy and paste the above URL into a browser to open the AWS "
                 "Management Console with limited permissions. When done, press "
                 "Enter to clean up and complete this demo.")
    finally:
        teardown(role)
        print("Thanks for watching!")
```

- For API details, see [AssumeRole in AWS SDK for Python \(Boto3\) API Reference](#).

## Get a session token that requires an MFA token with AWS STS using an AWS SDK

The following code example shows how to:

- Create an IAM role that grants permission to list Amazon S3 buckets.
- Create an IAM user that has permission to assume the role only when MFA credentials are provided.
- Register an MFA device for the user.
- Provide MFA credentials to get a session token and use temporary credentials to list S3 buckets.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an IAM user, register an MFA device, and create a role that grants permission to let the user list S3 buckets only when MFA credentials are used.

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual multi-factor authentication (MFA) device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates an inline policy for the user that lets the user list Amazon S3
    buckets,
    but only when MFA credentials are used.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
        that has permissions to create users, MFA devices, and
        policies in the account.
    :return: The newly created user, user key, and virtual MFA device.
    """
    user = iam_resource.create_user(UserName=unique_name('user'))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name('mfa'))
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(f"Showing the QR code for the device. Scan this in the MFA app of your "
          f"choice.")
    with open('qr.png', 'wb') as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
```

```

        AuthenticationCode2=mfa_code_2)
os.remove(qr_file.name)
print(f"MFA device is registered with the user.")

user_key = user.create_access_key_pair()
print(f"Created access key pair for user.")

print(f"Wait for user to be ready.", end='')
progress_bar(10)

user.create_policy(
    PolicyName=unique_name('user-policy'),
    PolicyDocument=json.dumps({
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Action': 's3>ListAllMyBuckets',
                'Resource': 'arn:aws:s3:::*',
                'Condition': {'Bool': {'aws:MultiFactorAuthPresent': True}}
            }
        ]
    })
)
print(f"Created an inline policy for {user.name} that lets the user list
buckets, "
      f"but only when MFA credentials are present.")

print("Give AWS time to propagate these new resources and connections.",
      end='')
progress_bar(10)

return user, user_key, virtual_mfa_device

```

Get temporary session credentials by passing an MFA token, and use the credentials to list S3 buckets for the account.

```

def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
MFA
                                device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp)
    else:
        response = sts_client.get_session_token()
    temp_credentials = response['Credentials']

    s3_resource = boto3.resource(
        's3',
        aws_access_key_id=temp_credentials['AccessKeyId'],
        aws_secret_access_key=temp_credentials['SecretAccessKey'],
        aws_session_token=temp_credentials['SessionToken'])

```

```
print(f"Buckets for the account:")
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

Destroy the resources created for the demo.

```
def teardown(user, virtual_mfa_device):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo MFA device.
    """
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

Run this scenario by using the previously defined functions.

```
def usage_demo():
    """
    Drives the demonstration.
    """
    print('*'*88)
    print(f"Welcome to the AWS Security Token Service assume role demo, "
          f"starring multi-factor authentication (MFA)!")
    print('*'*88)
    iam_resource = boto3.resource('iam')
    user, user_key, virtual_mfa_device = setup(iam_resource)
    try:
        sts_client = boto3.client(
            'sts', aws_access_key_id=user_key.id,
            aws_secret_access_key=user_key.secret)
        try:
            print("Listing buckets without specifying MFA credentials.")
            list_buckets_with_session_token_with_mfa(None, None, sts_client)
        except ClientError as error:
            if error.response['Error']['Code'] == 'AccessDenied':
                print("Got expected AccessDenied error.")
        mfa_totp = input('Enter the code from your registered MFA device: ')
        list_buckets_with_session_token_with_mfa(
            virtual_mfa_device.serial_number, mfa_totp, sts_client)
    finally:
        teardown(user, virtual_mfa_device)
    print("Thanks for watching!")
```

- For API details, see [GetSessionToken](#) in *AWS SDK for Python (Boto3) API Reference*.

# Code examples for SageMaker using AWS SDKs

The following code examples show how to use Amazon SageMaker with an AWS software development kit (SDK).

## Code examples

- [Actions for SageMaker using AWS SDKs \(p. 2107\)](#)
  - [Create a model in SageMaker using an AWS SDK \(p. 2107\)](#)
  - [Create a SageMaker endpoint using an AWS SDK \(p. 2108\)](#)
  - [Delete a model in SageMaker using an AWS SDK \(p. 2109\)](#)
  - [Delete a SageMaker endpoint using an AWS SDK \(p. 2110\)](#)
  - [Describe a SageMaker training job using an AWS SDK \(p. 2110\)](#)
  - [List models in SageMaker using an AWS SDK \(p. 2111\)](#)
  - [List the SageMaker notebook instances using an AWS SDK \(p. 2112\)](#)
  - [List the machine learning algorithms using an AWS SDK \(p. 2113\)](#)
  - [List SageMaker training jobs using an AWS SDK \(p. 2113\)](#)
  - [Create and start a SageMaker training job using an AWS SDK \(p. 2114\)](#)
  - [Create and start a SageMaker transform job using an AWS SDK \(p. 2117\)](#)
- [Scenarios for SageMaker using AWS SDKs \(p. 2118\)](#)
  - [Get started with SageMaker models and endpoints using an AWS SDK \(p. 2118\)](#)

## Actions for SageMaker using AWS SDKs

The following code examples show how to use Amazon SageMaker with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create a model in SageMaker using an AWS SDK \(p. 2107\)](#)
- [Create a SageMaker endpoint using an AWS SDK \(p. 2108\)](#)
- [Delete a model in SageMaker using an AWS SDK \(p. 2109\)](#)
- [Delete a SageMaker endpoint using an AWS SDK \(p. 2110\)](#)
- [Describe a SageMaker training job using an AWS SDK \(p. 2110\)](#)
- [List models in SageMaker using an AWS SDK \(p. 2111\)](#)
- [List the SageMaker notebook instances using an AWS SDK \(p. 2112\)](#)
- [List the machine learning algorithms using an AWS SDK \(p. 2113\)](#)
- [List SageMaker training jobs using an AWS SDK \(p. 2113\)](#)
- [Create and start a SageMaker training job using an AWS SDK \(p. 2114\)](#)
- [Create and start a SageMaker transform job using an AWS SDK \(p. 2117\)](#)

## Create a model in SageMaker using an AWS SDK

The following code example shows how to create a model in SageMaker.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_primarycontainer TYPE REF TO /aws1/cl_sgmcontainerdefn.

"Create an ABAP object for the container image based on input variables."
CREATE OBJECT lo_primarycontainer
  EXPORTING
    iv_image      = iv_container_image
    iv_modeldataurl = iv_model_data_url.

"Create a Sagemaker model"
TRY.
  oo_result = lo_sgm->createmodel(          " oo_result is returned for testing
purpose "
    iv_executionrolearn = iv_execution_role_arn
    iv_modelname = iv_model_name
    io_primarycontainer = lo_primarycontainer
  ).
  MESSAGE 'Model created' TYPE 'I'.
CATCH /aws1/cx_sgmresourcelimitexcd.
  MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateModel](#) in *AWS SDK for SAP ABAP API reference*.

## Create a SageMaker endpoint using an AWS SDK

The following code example shows how to create a SageMaker endpoint.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lt_production_variants TYPE /aws1/
cl_sgmproductionvariant=>tt_productionvariantlist.
DATA lo_production_variants TYPE REF TO /aws1/cl_sgmproductionvariant.
DATA oo_ep_config_result TYPE REF TO /aws1/cl_sgmcreateendptcfgout.
```

```

"Create a production variant as an ABAP object"
"Identifies a model that you want to host and the resources chosen to deploy
for hosting it."
CREATE OBJECT lo_production_variants
    EXPORTING
        iv_variantname      = iv_variant_name
        iv_modelname        = iv_model_name
        iv_initialinstancecount = iv_initial_instance_count
        iv_instancetype     = iv_instance_type.

    INSERT lo_production_variants INTO TABLE lt_production_variants.

"Create an endpoint configuration"
TRY.
    oo_ep_config_result = lo_sgm->createendpointconfig(
        iv_endpointconfigname = iv_endpoint_config_name
        it_productionvariants = lt_production_variants
    ).
    MESSAGE 'Endpoint configuration created' TYPE 'I'.
CATCH /aws1/cx_sgmresourcelimitexcd.
    MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
ENDTRY.

"Create an endpoint"
TRY.
    oo_result = lo_sgm->createendpoint(      " oo_result is returned for testing
purpose "
        iv_endpointconfigname = iv_endpoint_config_name
        iv_endpointname       = iv_endpoint_name
    ).
    MESSAGE 'Endpoint created' TYPE 'I'.
CATCH /aws1/cx_sgmresourcelimitexcd.
    MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
ENDTRY.

```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [CreateEndpoint](#)
  - [CreateEndpointConfig](#)

## Delete a model in SageMaker using an AWS SDK

The following code example shows how to delete a model in SageMaker.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
    lo_sgm->deletemodel(
        iv_modelname = iv_model_name
    ).
```

```
MESSAGE 'Model deleted' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
  DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
  MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DeleteModel](#) in *AWS SDK for SAP ABAP API reference*.

## Delete a SageMaker endpoint using an AWS SDK

The following code example shows how to delete a SageMaker endpoint.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Delete an endpoint"
TRY.
  lo_sgm->deleteendpoint(
    iv_endpointname = iv_endpoint_name
  ).
  MESSAGE 'Endpoint configuration deleted' TYPE 'I'.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpoint_exception).
    DATA(lv_endpoint_error) = |"{ lo_endpoint_exception->av_err_code }" - 
{ lo_endpoint_exception->av_err_msg }|.
    MESSAGE lv_endpoint_error TYPE 'E'.
ENDTRY.

"Delete an endpoint configuration"
TRY.
  lo_sgm->deleteendpointconfig(
    iv_endpointconfigname = iv_endpoint_config_name
  ).
  MESSAGE 'Endpoint deleted' TYPE 'I'.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpointconfig_exception).
    DATA(lv_endpointconfig_error) = |"{ lo_endpointconfig_exception-
>av_err_code }" - { lo_endpointconfig_exception->av_err_msg }|.
    MESSAGE lv_endpointconfig_error TYPE 'E'.
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [DeleteEndpoint](#)
  - [DeleteEndpointConfig](#)

## Describe a SageMaker training job using an AWS SDK

The following code example shows how to describe a SageMaker training job.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sgm->describetrainingjob(          " oo_result is returned for  
testing purpose "  
        iv_trainingjobname = iv_training_job_name  
    ).  
    MESSAGE 'Retrieved description of training job' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.  
        MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [DescribeTrainingJob](#) in *AWS SDK for SAP ABAP API reference*.

## List models in SageMaker using an AWS SDK

The following code example shows how to list models in SageMaker.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sgm->listmodels(          " oo_result is returned for  
testing purpose "  
        iv_namecontains = iv_name_contains  
    ).  
    MESSAGE 'Retrieved list of models' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.  
        MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [ListModels](#) in *AWS SDK for SAP ABAP API reference*.

## List the SageMaker notebook instances using an AWS SDK

The following code examples show how to list SageMaker notebook instances.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_instances(client: &Client) -> Result<(), Error> {
    let notebooks = client.list_notebook_instances().send().await?;

    println!("Notebooks:");

    for n in notebooks.notebook_instances().unwrap_or_default() {
        let n_instance_type = n.instance_type().unwrap();
        let n_status = n.notebook_instance_status().unwrap();
        let n_name = n.notebook_instance_name().unwrap_or_default();

        println!("  Name : {}", n_name);
        println!("  Status : {}", n_status.as_ref());
        println!("  Instance Type : {}", n_instance_type.as_ref());
        println!();
    }

    Ok(())
}
```

- For API details, see [ListNotebookInstances](#) in *AWS SDK for Rust API reference*.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_sgm->listnotebookinstances(          " oo_result is returned
for testing purpose "
        iv_namecontains = iv_name_contains
    ).
    MESSAGE 'Retrieved list of notebook instances' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [ListNotebookInstances](#) in *AWS SDK for SAP ABAP API reference*.

## List the machine learning algorithms using an AWS SDK

The following code example shows how to list SageMaker machine learning algorithms.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sgm->listalgorithms(           " oo_result is returned for  
testing purpose "  
        iv_namecontains = iv_name_contains  
    ).  
    MESSAGE 'Retrieved list of algorithms' TYPE 'I'.  
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- For API details, see [ListAlgorithms](#) in *AWS SDK for SAP ABAP API reference*.

## List SageMaker training jobs using an AWS SDK

The following code examples show how to list SageMaker training jobs.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_jobs(client: &Client) -> Result<(), Error> {  
    let job_details = client.list_training_jobs().send().await?;  
  
    println!("Jobs:");  
  
    for j in job_details.training_job_summaries().unwrap_or_default() {  
        let name = j.training_job_name().unwrap_or_default();  
        let creation_time = j.creation_time().unwrap().to_chrono_utc();  
    }  
}
```

```
let training_end_time = j.training_end_time().unwrap().to_chrono_utc();

let status = j.training_job_status().unwrap();
let duration = training_end_time - creation_time;

println!("  Name:          {}", name);
println!(
    "  Creation date/time: {}",
    creation_time.format("%Y-%m-%d@%H:%M:%S")
);
println!("  Duration (seconds): {}", duration.num_seconds());
println!("  Status:         {}", status.as_ref());

println!();
}

Ok(())
}
```

- For API details, see [ListTrainingJobs](#) in *AWS SDK for Rust API reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  oo_result = lo_sgm->listtrainingjobs(           " oo_result is returned for
testing purpose "
    iv_namecontains = iv_name_contains
    iv_maxresults = iv_max_results
  ).
  MESSAGE 'Retrieved list of training jobs' TYPE 'I'.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [ListTrainingJobs](#) in *AWS SDK for SAP ABAP API reference*.

## Create and start a SageMaker training job using an AWS SDK

The following code example shows how to start a SageMaker training job.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_hyperparameters_w TYPE REF TO /aws1/cl_sgmhyperparameters_w.
DATA lt_hyperparameters TYPE /aws1/cl_sgmhyperparameters_w=>tt_hyperparameters.
DATA lt_input_data_config TYPE /aws1/cl_sgmcchannel=>tt_inputdataconfig.
DATA lo_trn_channel TYPE REF TO /aws1/cl_sgmcchannel.
DATA lo_trn_datasource TYPE REF TO /aws1/cl_sgmsdatasource.
DATA lo_trn_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
DATA lo_val_channel TYPE REF TO /aws1/cl_sgmcchannel.
DATA lo_val_datasource TYPE REF TO /aws1/cl_sgmsdatasource.
DATA lo_val_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
DATA lo_algorithm_specification TYPE REF TO /aws1/cl_sgmalgorithmspec.
DATA lo_resource_config TYPE REF TO /aws1/cl_sgmresourceconfig.
DATA lo_output_data_config TYPE REF TO /aws1/cl_sgmooutputdataconfig.
DATA lo_stopping_condition TYPE REF TO /aws1/cl_sgmoStoppingCondition.

"Create ABAP internal table for hyperparameters based on input variables."
"These hyperparameters are based on Amazon SageMaker built-in algorithm - XGBoost"
CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_max_depth.
INSERT VALUE #( key = 'max_depth' value = lo_hyperparameters_w ) INTO TABLE lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_eta.
INSERT VALUE #( key = 'eta' value = lo_hyperparameters_w ) INTO TABLE lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_eval_metric.
INSERT VALUE #( key = 'eval_metric' value = lo_hyperparameters_w ) INTO TABLE lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_scale_pos_weight.
INSERT VALUE #( key = 'scale_pos_weight' value = lo_hyperparameters_w ) INTO TABLE lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_subsample.
INSERT VALUE #( key = 'subsample' value = lo_hyperparameters_w ) INTO TABLE lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_objective.
INSERT VALUE #( key = 'objective' value = lo_hyperparameters_w ) INTO TABLE lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_num_round.
INSERT VALUE #( key = 'num_round' value = lo_hyperparameters_w ) INTO TABLE lt_hyperparameters.

"Create ABAP objects for training data sources"
CREATE OBJECT lo_trn_s3datasource
EXPORTING
    iv_s3datatype           = iv_trn_data_s3datatype
    iv_s3datadistributiontype = iv_trn_data_s3datadistribution
    iv_s3uri                 = iv_trn_data_s3uri.

CREATE OBJECT lo_trn_datasource
EXPORTING
    io_s3datasource = lo_trn_s3datasource.

CREATE OBJECT lo_trn_channel
EXPORTING
    iv_channelname      = 'train'
```

```

        io_datasource      = lo_trn_datasource
        iv_compressiontype = iv_trn_data_compressiontype
        iv_contenttype     = iv_trn_data_contenttype.

INSERT lo_trn_channel INTO TABLE lt_input_data_config.

"Create ABAP objects for validation data sources"
CREATE OBJECT lo_val_s3datasource
    EXPORTING
        iv_s3datatype      = iv_val_data_s3datatype
        iv_s3datadistributiontype = iv_val_data_s3datadistribution
        iv_s3uri           = iv_val_data_s3uri.

CREATE OBJECT lo_val_datasource
    EXPORTING
        io_s3datasource = lo_val_s3datasource.

CREATE OBJECT lo_val_channel
    EXPORTING
        iv_channelname     = 'validation'
        io_datasource      = lo_val_datasource
        iv_compressiontype = iv_val_data_compressiontype
        iv_contenttype     = iv_val_data_contenttype.

INSERT lo_val_channel INTO TABLE lt_input_data_config.

"Create an ABAP object for algorithm specification."
CREATE OBJECT lo_algorithm_specification
    EXPORTING
        iv_trainingimage    = iv_training_image
        iv_traininginputmode = iv_training_input_mode.

"Create an ABAP object for resource configuration"
CREATE OBJECT lo_resource_config
    EXPORTING
        iv_instancecount   = iv_instance_count
        iv_instancetype    = iv_instance_type
        iv_volumesizeingb = iv_volume_sizeingb.

"Create an ABAP object for output data configuration"
CREATE OBJECT lo_output_data_config
    EXPORTING
        iv_s3outputpath = iv_s3_output_path.

"Create ABAP object for stopping condition"
CREATE OBJECT lo_stopping_condition
    EXPORTING
        iv_maxruntimeinseconds = iv_max_runtime_in_seconds.

"Create a training job"
TRY.
    oo_result = lo_sgm->createtrainingjob(      " oo_result is returned for
testing purpose "
        iv_trainingjobname      = iv_training_job_name
        iv_rolearn                = iv_role_arn
        it_hyperparameters       = lt_hyperparameters
        it_inputdataconfig        = lt_input_data_config
        io_algorithmspecification = lo_algorithm_specification
        io_outputdataconfig       = lo_output_data_config
        io_resourceconfig         = lo_resource_config
        io_stoppingcondition     = lo_stopping_condition
    ).
    MESSAGE 'Training job created' TYPE 'I'.
CATCH /aws1/cx_sgmresourceinuse.
    MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcenotfound.

```

```
MESSAGE 'Resource being access is not found.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcelimitexcd.
MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateTrainingJob](#) in *AWS SDK for SAP ABAP API reference*.

## Create and start a SageMaker transform job using an AWS SDK

The following code example shows how to start a SageMaker transform job.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_transforminput TYPE REF TO /aws1/cl_sgmtransforminput.
DATA lo_transformoutput TYPE REF TO /aws1/cl_sgmtransformoutput.
DATA lo_transformresources TYPE REF TO /aws1/cl_sgmtransformresources.
DATA lo_datasource  TYPE REF TO /aws1/cl_sgmtransformdatasrc.
DATA lo_s3datasource  TYPE REF TO /aws1/cl_sgmtransforms3datasrc.

"Create ABAP object for S3 data source"
CREATE OBJECT lo_s3datasource
  EXPORTING
    iv_s3uri      = iv_tf_data_s3uri
    iv_s3datatype = iv_tf_data_s3datatype.

"Create ABAP object for data source"
CREATE OBJECT lo_datasource
  EXPORTING
    io_s3datasource = lo_s3datasource.

"Create ABAP object for transform data source."
CREATE OBJECT lo_transforminput
  EXPORTING
    io_datasource     = lo_datasource
    iv_contenttype   = iv_tf_data_contenttype
    iv_compressiontype = iv_tf_data_compressiontype.

"Create an ABAP object for resource configuration"
CREATE OBJECT lo_transformresources
  EXPORTING
    iv_instancecount = iv_instance_count
    iv_instancetype  = iv_instance_type.

"Create an ABAP object for output data configuration"
CREATE OBJECT lo_transformoutput
  EXPORTING
    iv_s3outputpath = iv_s3_output_path.

"Create a transform job"
TRY.
```

```
    oo_result = lo_sgm->createtransformjob(      " oo_result is returned for
testing purpose "
        iv_modelname = iv_tf_model_name
        iv_transformjobname = iv_tf_job_name
        io_transforminput = lo_transforminput
        io_transformoutput = lo_transformoutput
        io_transformresources = lo_transformresources
    ).
    MESSAGE 'Transform job created' TYPE 'I'.
CATCH /aws1/cx_sgmresourceinuse.
    MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcenotfound.
    MESSAGE 'Resource being access is not found.' TYPE 'E'.
CATCH /aws1/cx_sgmresource limitecd.
    MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateTransformJob](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios for SageMaker using AWS SDKs

The following code examples show how to use Amazon SageMaker with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Get started with SageMaker models and endpoints using an AWS SDK \(p. 2118\)](#)

## Get started with SageMaker models and endpoints using an AWS SDK

The following code example shows how to:

- Start a training job.
- Create a SageMaker model.
- Create an endpoint configuration.
- Create an endpoint.
- Delete an endpoint.
- Delete an endpoint configuration.
- Delete a model.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_hyperparameters_w TYPE REF TO /aws1/cl_sgmhyperparameters_w.
```

```

DATA lo_trn_channel TYPE REF TO /aws1/cl_sgmchannel.
DATA lo_trn_datasource TYPE REF TO /aws1/cl_sgmdatasource.
DATA lo_trn_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
DATA lo_val_channel TYPE REF TO /aws1/cl_sgmchannel.
DATA lo_val_datasource TYPE REF TO /aws1/cl_sgmdatasource.
DATA lo_val_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
DATA lo_algorithm_specification TYPE REF TO /aws1/cl_sgmalgorithmspec.
DATA lo_resource_config TYPE REF TO /aws1/cl_sgmresourceconfig.
DATA lo_output_data_config TYPE REF TO /aws1/cl_sgmooutputdataconfig.
DATA lo_stopping_condition TYPE REF TO /aws1/cl_sgmooutputdataconfig.
DATA lo_primarycontainer TYPE REF TO /aws1/cl_sgmcontainerdefn.
DATA lo_production_variants TYPE REF TO /aws1/cl_sgmproductionvariant.
DATA lo_ep_config_result TYPE REF TO /aws1/cl_sgmcfgout.
DATA lo_training_result TYPE REF TO /aws1/cl_sgmdescirtrnjobrsp.
DATA lt_production_variants TYPE /aws1/
cl_sgmproductionvariant=>tt_productionvariantlist.
DATA lt_input_data_config TYPE /aws1/cl_sgmchannel=>tt_inputdataconfig.
DATA lt_hyperparameters TYPE /aws1/cl_sgmyhyperparameters_w=>tt_hyperparameters.
DATA lv_model_data_url TYPE /aws1/sgmurl.

lv_model_data_url = iv_s3_output_path && iv_training_job_name && '/output/
model.tar.gz'.

"Create ABAP internal table for hyperparameters based on input variables."
"These hyperparameters are based on Amazon SageMaker built-in algorithm -
XGBoost"
CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_max_depth.
INSERT VALUE #( key = 'max_depth' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_eta.
INSERT VALUE #( key = 'eta' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_eval_metric.
INSERT VALUE #( key = 'eval_metric' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_scale_pos_weight.
INSERT VALUE #( key = 'scale_pos_weight' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_subsample.
INSERT VALUE #( key = 'subsample' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_objective.
INSERT VALUE #( key = 'objective' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_num_round.
INSERT VALUE #( key = 'num_round' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

"Create ABAP internal table for data based on input variables."
"Training data"
CREATE OBJECT lo_trn_s3datasource
EXPORTING
    iv_s3datatype           = iv_trn_data_s3datatype
    iv_s3datadistributiontype = iv_trn_data_s3datadistribution
    iv_s3uri                 = iv_trn_data_s3uri.

CREATE OBJECT lo_trn_datasource EXPORTING io_s3datasource =
lo_trn_s3datasource.

CREATE OBJECT lo_trn_channel

```

```

EXPORTING
    iv_channelname      = 'train'
    io_datasource       = lo_trn_datasource
    iv_compressiontype = iv_trn_data_compressiontype
    iv_contenttype     = iv_trn_data_contenttype.
INSERT lo_trn_channel INTO TABLE lt_input_data_config.

"Validation data"
CREATE OBJECT lo_val_s3datasource
    EXPORTING
        iv_s3datatype      = iv_val_data_s3datatype
        iv_s3datadistributiontype = iv_val_data_s3datadistribution
        iv_s3uri           = iv_val_data_s3uri.

CREATE OBJECT lo_val_datasource EXPORTING io_s3datasource =
lo_val_s3datasource.

CREATE OBJECT lo_val_channel
    EXPORTING
        iv_channelname     = 'validation'
        io_datasource      = lo_val_datasource
        iv_compressiontype = iv_val_data_compressiontype
        iv_contenttype     = iv_val_data_contenttype.
INSERT lo_val_channel INTO TABLE lt_input_data_config.

"Create an ABAP object for algorithm specification based on input variables."
CREATE OBJECT lo_algorithm_specification
    EXPORTING
        iv_trainingimage    = iv_training_image
        iv_traininginputmode = iv_training_input_mode.

"Create an ABAP object for resource configuration"
CREATE OBJECT lo_resource_config
    EXPORTING
        iv_instancecount   = iv_instance_count
        iv_instancetype    = iv_instance_type
        iv_volumesizeingb = iv_volume_sizeingb.

"Create an ABAP object for output data configuration"
CREATE OBJECT lo_output_data_config EXPORTING iv_s3outputpath =
iv_s3_output_path.

"Create ABAP object for stopping condition"
CREATE OBJECT lo_stopping_condition EXPORTING iv_maxruntimeinseconds =
iv_max_runtime_in_seconds.

TRY.
    lo_sgm->createtrainingjob(
        iv_trainingjobname      = iv_training_job_name
        iv_rolearn               = iv_role_arn
        it_hyperparameters       = lt_hyperparameters
        it_inputdataconfig       = lt_input_data_config
        io_algorithmspecification = lo_algorithm_specification
        io_outputdataconfig      = lo_output_data_config
        io_resourceconfig        = lo_resource_config
        io_stoppingcondition     = lo_stopping_condition
    ).
    MESSAGE 'Training job created' TYPE 'I'.
CATCH /aws1/cx_sgmresourceinuse.
    MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcenotfound.
    MESSAGE 'Resource being access is not found.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcelimitexcd.
    MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
ENDTRY.

```

```

"Wait for training job to be completed"
lo_training_result = lo_sgm->describetrainingjob( iv_trainingjobname =
iv_training_job_name ).
WHILE lo_training_result->get_trainingjobstatus( ) <> 'Completed'.
  IF sy-index = 30.
    EXIT.                      "maximum 900 seconds"
  ENDIF.
  WAIT UP TO 30 SECONDS.
  lo_training_result = lo_sgm->describetrainingjob( iv_trainingjobname =
iv_training_job_name ).
ENDWHILE.

"Create ABAP object for the container image based on input variables."
CREATE OBJECT lo_primarycontainer
  EXPORTING
    iv_image      = iv_training_image
    iv_modeldataurl = lv_model_data_url.

"Create a Sagemaker model"
TRY.
  lo_sgm->createmodel(
    iv_executionrolearn = iv_role_arn
    iv_modelname = iv_model_name
    io_primarycontainer = lo_primarycontainer
  ).
  MESSAGE 'Model created' TYPE 'I'.
CATCH /aws1/cx_sgmresourcelimitexcd.
  MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
ENDTRY.

"Create an endpoint production variant"
CREATE OBJECT lo_production_variants
  EXPORTING
    iv_variantname      = iv_ep_variant_name
    iv_modelname        = iv_model_name
    iv_initialinstancecount = iv_ep_initial_instance_count
    iv_instancetype     = iv_ep_instance_type.
  INSERT lo_production_variants INTO TABLE lt_production_variants.

TRY.
  "Create an endpoint configuration"
  lo_ep_config_result = lo_sgm->createendpointconfig(
    iv_endpointconfigname = iv_ep_cfg_name
    it_productionvariants = lt_production_variants
  ).
  MESSAGE 'Endpoint configuration created' TYPE 'I'.

  "Create an endpoint"
  oo_ep_output = lo_sgm->createendpoint(          " oo_ep_output is returned
for testing purpose "
    iv_endpointconfigname = iv_ep_cfg_name
    iv_endpointname = iv_ep_name
  ).
  MESSAGE 'Endpoint created' TYPE 'I'.
CATCH /aws1/cx_sgmresourcelimitexcd.
  MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
ENDTRY.

"Wait for endpoint creation to be completed"
DATA(lo_endpoint_result) = lo_sgm->describeendpoint( iv_endpointname =
iv_ep_name ).
WHILE lo_endpoint_result->get_endpointstatus( ) <> 'InService'.
  IF sy-index = 30.
    EXIT.                      "maximum 900 seconds"
  ENDIF.
  WAIT UP TO 30 SECONDS.

```

```
    lo_endpoint_result = lo_sgm->describeendpoint( iv_endpointname =
iv_ep_name ).
ENDWHILE.

TRY.
    "Delete an endpoint"
    lo_sgm->deleteendpoint(
        iv_endpointname = iv_ep_name
    ).
MESSAGE 'Endpoint deleted' TYPE 'I'.

    "Delete an endpoint configuration"
    lo_sgm->deleteendpointconfig(
        iv_endpointconfigname = iv_ep_cfg_name
    ).
MESSAGE 'Endpoint configuration deleted' TYPE 'I'.

    "Delete model"
    lo_sgm->deletemodel(
        iv_modelname = iv_model_name
    ).
MESSAGE 'Model deleted' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpointconfig_exception).
    DATA(lv_endpointconfig_error) = |"{ lo_endpointconfig_exception-
>av_err_code }" - { lo_endpointconfig_exception->av_err_msg }|.
    MESSAGE lv_endpointconfig_error TYPE 'E'.
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [CreateEndpoint](#)
  - [CreateEndpointConfig](#)
  - [CreateModel](#)
  - [CreateTrainingJob](#)
  - [DeleteEndpoint](#)
  - [DeleteEndpointConfig](#)
  - [DeleteModel](#)
  - [DescribeEndpoint](#)
  - [DescribeTrainingJob](#)

## Code examples for Secrets Manager using AWS SDKs

The following code examples show how to use AWS Secrets Manager with an AWS software development kit (SDK).

### Code examples

- [Actions for Secrets Manager using AWS SDKs \(p. 2123\)](#)
  - [Create a Secrets Manager secret using an AWS SDK \(p. 2123\)](#)
  - [Delete a Secrets Manager secret using an AWS SDK \(p. 2126\)](#)
  - [Describe a Secrets Manager secret using an AWS SDK \(p. 2127\)](#)
  - [Get a random password from Secrets Manager using an AWS SDK \(p. 2129\)](#)
  - [Get a Secrets Manager secret value using an AWS SDK \(p. 2130\)](#)
  - [List Secrets Manager secrets using an AWS SDK \(p. 2135\)](#)

- Put a value in a Secrets Manager secret using an AWS SDK (p. 2137)
- Update the stage of a Secrets Manager secret version using an AWS SDK (p. 2139)
- Scenarios for Secrets Manager using AWS SDKs (p. 2140)
  - Create and manage a Secrets Manager secret using an AWS SDK (p. 2140)
- Cross-service examples for Secrets Manager using AWS SDKs (p. 2141)
  - Create a lending library REST API (p. 2141)

## Actions for Secrets Manager using AWS SDKs

The following code examples show how to use AWS Secrets Manager with AWS SDKs. Each example calls an individual service function.

### Examples

- Create a Secrets Manager secret using an AWS SDK (p. 2123)
- Delete a Secrets Manager secret using an AWS SDK (p. 2126)
- Describe a Secrets Manager secret using an AWS SDK (p. 2127)
- Get a random password from Secrets Manager using an AWS SDK (p. 2129)
- Get a Secrets Manager secret value using an AWS SDK (p. 2130)
- List Secrets Manager secrets using an AWS SDK (p. 2135)
- Put a value in a Secrets Manager secret using an AWS SDK (p. 2137)
- Update the stage of a Secrets Manager secret version using an AWS SDK (p. 2139)

## Create a Secrets Manager secret using an AWS SDK

The following code examples show how to create a Secrets Manager secret.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
int main(int argc, const char *argv[])
{
    if (argc != 3) {
        std::cout << "Usage:\n" <<
        "    <secretName> <secretValue> \n\n" <<
        "Where:\n" <<
        "    secretName - The name of the secret (for example, tutorials/
MyFirstSecret). \n" <<
        "    secretValue - The secret value. " << std::endl;
        return 0;
    }

    SDKOptions options;
    options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;

    InitAPI(options);
    {
        Aws::Client::ClientConfiguration config;

        //TODO(user): Enter the Region where you want to create the secret.
    }
}
```

```
String region = "us-east-1";
if (!region.empty())
{
    config.region = region;
}
SecretsManager::SecretsManagerClient sm_client(config);

String secretName = argv[1];
String secretString = argv[2];
SecretsManager::Model::CreateSecretRequest request;
request.SetName(secretName);
request.SetSecretString(secretString);

auto createSecretOutcome = sm_client.CreateSecret(request);
if(createSecretOutcome.IsSuccess()){
    std::cout << "Create secret with name: " <<
createSecretOutcome.GetResult().GetName() << std::endl;
} else{
    std::cout << "Failed with Error: " <<
createSecretOutcome.GetError() << std::endl;
}
}

ShutdownAPI(options);
return 0;
}
```

- For API details, see [CreateSecret](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createNewSecret( SecretsManagerClient secretsClient,
String secretName, String secretValue) {

    try {
        CreateSecretRequest secretRequest = CreateSecretRequest.builder()
            .name(secretName)
            .description("This secret was created by the AWS Secret Manager
Java API")
            .secretString(secretValue)
            .build();

        CreateSecretResponse secretResponse =
secretsClient.createSecret(secretRequest);
        return secretResponse.arn();

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateSecret](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createNewSecret(secretName: String?, secretValue: String?): String? {  
  
    val request = CreateSecretRequest {  
        name = secretName  
        description = "This secret was created by the AWS Secrets Manager Kotlin  
API"  
        secretString = secretValue  
    }  
  
    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->  
        val response = secretsClient.createSecret(request)  
        return response.arn  
    }  
}
```

- For API details, see [CreateSecret](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:  
    """Encapsulates Secrets Manager functions."""  
    def __init__(self, secretsmanager_client):  
        """  
        :param secretsmanager_client: A Boto3 Secrets Manager client.  
        """  
        self.secretsmanager_client = secretsmanager_client  
        self.name = None  
  
    def create(self, name, secret_value):  
        """  
        Creates a new secret. The secret value can be a string or bytes.  
  
        :param name: The name of the secret to create.  
        :param secret_value: The value of the secret.  
        :return: Metadata about the newly created secret.  
        """  
        self._clear()  
        try:  
            kwargs = {'Name': name}  
            if isinstance(secret_value, str):  
                kwargs['SecretString'] = secret_value  
            elif isinstance(secret_value, bytes):  
                kwargs['SecretBinary'] = secret_value
```

```
response = self.secretsmanager_client.create_secret(**kwargs)
self.name = name
logger.info("Created secret %s.", name)
except ClientError:
    logger.exception("Couldn't get secret %s.", name)
    raise
else:
    return response
```

- For API details, see [CreateSecret](#) in *AWS SDK for Python (Boto3) API Reference*.

## Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_secret(client: &Client, name: &str, value: &str) -> Result<(), Error>
{
    client
        .create_secret()
        .name(name)
        .secret_string(value)
        .send()
        .await?;

    println!("Created secret");

    Ok(())
}
```

- For API details, see [CreateSecret](#) in *AWS SDK for Rust API reference*.

## Delete a Secrets Manager secret using an AWS SDK

The following code examples show how to delete a Secrets Manager secret.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteSpecificSecret(SecretsManagerClient secretsClient,
String secretName) {

    try {
        DeleteSecretRequest secretRequest = DeleteSecretRequest.builder()
            .secretId(secretName)
```

```
        .build();

        secretsClient.deleteSecret(secretRequest);
        System.out.println(secretName + " is deleted.");

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteSecret](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:
    """Encapsulates Secrets Manager functions."""
    def __init__(self, secretsmanager_client):
        """
        :param secretsmanager_client: A Boto3 Secrets Manager client.
        """
        self.secretsmanager_client = secretsmanager_client
        self.name = None

    def delete(self, without_recovery):
        """
        Deletes the secret.

        :param without_recovery: Permanently deletes the secret immediately when
        True; otherwise, the deleted secret can be restored
        within
                    the recovery window. The default recovery window
        is
                    30 days.
        """
        if self.name is None:
            raise ValueError

        try:
            self.secretsmanager_client.delete_secret(
                SecretId=self.name, ForceDeleteWithoutRecovery=without_recovery)
            logger.info("Deleted secret %s.", self.name)
            self._clear()
        except ClientError:
            logger.exception("Deleted secret %s.", self.name)
            raise
```

- For API details, see [DeleteSecret](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a Secrets Manager secret using an AWS SDK

The following code examples show how to describe a Secrets Manager secret.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeGivenSecret(SecretsManagerClient secretsClient,
String secretName) {

    try {
        DescribeSecretRequest secretRequest = DescribeSecretRequest.builder()
            .secretId(secretName)
            .build();

        DescribeSecretResponse secretResponse =
secretsClient.describeSecret(secretRequest);
        Instant lastChangedDate = secretResponse.lastChangedDate();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
            DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )
                .withLocale( Locale.US )
                .withZone( ZoneId.systemDefault() );

        formatter.format( lastChangedDate );
        System.out.println("The date of the last change to "+
secretResponse.name() +" is " + lastChangedDate );

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeSecret](#) in [AWS SDK for Java 2.x API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeGivenSecret(secretName: String?) {

    val secretRequest = DescribeSecretRequest {
        secretId = secretName
    }

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        val response = secretsClient.describeSecret(secretRequest)
        val secArn = response.description
```

```
        }     println("The secret description is $secArn")
    }
```

- For API details, see [DescribeSecret](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:
    """Encapsulates Secrets Manager functions."""
    def __init__(self, secretsmanager_client):
        """
        :param secretsmanager_client: A Boto3 Secrets Manager client.
        """
        self.secretsmanager_client = secretsmanager_client
        self.name = None

    def describe(self, name=None):
        """
        Gets metadata about a secret.

        :param name: The name of the secret to load. If `name` is None, metadata
        about
            the current secret is retrieved.
        :return: Metadata about the secret.
        """
        if self.name is None and name is None:
            raise ValueError
        if name is None:
            name = self.name
        self._clear()
        try:
            response = self.secretsmanager_client.describe_secret(SecretId=name)
            self.name = name
            logger.info("Got secret metadata for %s.", name)
        except ClientError:
            logger.exception("Couldn't get secret metadata for %s.", name)
            raise
        else:
            return response
```

- For API details, see [DescribeSecret](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a random password from Secrets Manager using an AWS SDK

The following code example shows how to get a random password from Secrets Manager.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:  
    """Encapsulates Secrets Manager functions."""  
    def __init__(self, secretsmanager_client):  
        """  
        :param secretsmanager_client: A Boto3 Secrets Manager client.  
        """  
        self.secretsmanager_client = secretsmanager_client  
        self.name = None  
  
    def get_random_password(self, pw_length):  
        """  
        Gets a randomly generated password.  
  
        :param pw_length: The length of the password.  
        :return: The generated password.  
        """  
        try:  
            response = self.secretsmanager_client.get_random_password(  
                PasswordLength=pw_length)  
            password = response['RandomPassword']  
            logger.info("Got random password.")  
        except ClientError:  
            logger.exception("Couldn't get random password.")  
            raise  
        else:  
            return password
```

- For API details, see [GetRandomPassword in AWS SDK for Python \(Boto3\) API Reference](#).

## Get a Secrets Manager secret value using an AWS SDK

The following code examples show how to get a Secrets Manager secret value.

### .NET

#### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.IO;  
using System.Threading.Tasks;  
using Amazon.SecretsManager;  
using Amazon.SecretsManager.Model;  
  
/// <summary>  
/// This example uses the Amazon Web Service Secrets Manager to retrieve  
/// the secret value for the provided secret name. This example was created
```

```
/// using the AWS SDK for .NET v3.7 and .NET Core 5.0.
/// </summary>
public class GetSecretValue
{
    /// <summary>
    /// The main method initializes the necessary values and then calls
    /// the GetSecretAsync and DecodeString methods to get the decoded
    /// secret value for the secret named in secretName.
    /// </summary>
    public static async Task Main()
    {
        string secretName = "<<{{MySecretName}}>>";
        string secret;

        IAmazonSecretsManager client = new AmazonSecretsManagerClient();

        var response = await GetSecretAsync(client, secretName);

        if (response is not null)
        {
            secret = DecodeString(response);

            if (!string.IsNullOrEmpty(secret))
            {
                Console.WriteLine($"The decoded secret value is: {secret}.");
            }
            else
            {
                Console.WriteLine("No secret value was returned.");
            }
        }
    }

    /// <summary>
    /// Retrieves the secret value given the name of the secret to
    /// retrieve.
    /// </summary>
    /// <param name="client">The client object used to retrieve the secret
    /// value for the given secret name.</param>
    /// <param name="secretName">The name of the secret value to retrieve.</param>
    /// <returns>The GetSecretValueReponse object returned by
    /// GetSecretValueAsync.</returns>
    public static async Task<GetSecretValueResponse> GetSecretAsync(
        IAmazonSecretsManager client,
        string secretName)
    {
        GetSecretValueRequest request = new();
        request.SecretId = secretName;
        request.VersionStage = "AWSCURRENT"; // VersionStage defaults to
        AWSCURRENT if unspecified.

        GetSecretValueResponse response = null;

        // For the sake of simplicity, this example handles only the most
        // general SecretsManager exception.
        try
        {
            response = await client.GetSecretValueAsync(request);
        }
        catch (AmazonSecretsManagerException e)
        {
            Console.WriteLine($"Error: {e.Message}");
        }
    }

    return response;
}
```

```
}

/// <summary>
/// Decodes the secret returned by the call to GetSecretValueAsync and
/// returns it to the calling program.
/// </summary>
/// <param name="response">A GetSecretValueResponse object containing
/// the requested secret value returned by GetSecretValueAsync.</param>
/// <returns>A string representing the decoded secret value.</returns>
public static string DecodeString(GetSecretValueResponse response)
{
    // Decrypts secret using the associated AWS Key Management Service
    // Customer Master Key (CMK.) Depending on whether the secret is a
    // string or binary value, one of these fields will be populated.
    MemoryStream memoryStream = new();

    if (response.SecretString is not null)
    {
        var secret = response.SecretString;
        return secret;
    }
    else if (response.SecretBinary is not null)
    {
        memoryStream = response.SecretBinary;
        StreamReader reader = new StreamReader(memoryStream);
        string decodedBinarySecret =
System.Text.Encoding.UTF8.GetString(Convert.FromBase64String(reader.ReadToEnd()));
        return decodedBinarySecret;
    }
    else
    {
        return string.Empty;
    }
}
```

- For API details, see [GetSecretValue](#) in *AWS SDK for .NET API Reference*.

## C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
int main(int argc, const char *argv[])
{
    if (argc != 2) {
        std::cout << "Usage:\n" <<
                    "    <secretName> \n\n" <<
                    "Where:\n" <<
                    "    secretName - The name of the secret (for example, tutorials/
MyFirstSecret). \n"
                    << std::endl;
        return 0;
    }

    SDKOptions options;
    options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
```

```
InitAPI(options);
{
    Aws::Client::ClientConfiguration config;

    //TODO(user): Enter the Region where you want to create the secret.
    String region = "us-east-1";
    if (!region.empty())
    {
        config.region = region;
    }
    SecretsManager::SecretsManagerClient sm_client(config);

    String secretId = argv[1];
    SecretsManager::Model::GetSecretValueRequest request;
    request.SetSecretId(secretId);

    auto getSecretValueOutcome = sm_client.GetSecretValue(request);
    if(getSecretValueOutcome.IsSuccess()){
        std::cout << "Secret is: " <<
        getSecretValueOutcome.GetResult().GetSecretString() << std::endl;
    }else{
        std::cout << "Failed with Error: " <<
        getSecretValueOutcome.GetError() << std::endl;
    }
}

ShutdownAPI(options);
return 0;
}
```

- For API details, see [GetSecretValue](#) in *AWS SDK for C++ API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getValue(SecretsManagerClient secretsClient, String
secretName) {

    try {
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
secretsClient.getSecretValue(valueRequest);
        String secret = valueResponse.secretString();
        System.out.println(secret);

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [GetSecretValue](#) in *AWS SDK for Java 2.x API Reference*.

Kotlin

SDK for Kotlin

## Note

This is prerelease documentation for a feature in preview release. It is subject to change.

## Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    suspend fun getValue(secretName: String?) {  
  
        val valueRequest = GetSecretValueRequest {  
            secretId = secretName  
        }  
  
        SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->  
            val response = secretsClient.getSecretValue(valueRequest)  
            val secret = response.secretString  
            println("The secret value is $secret")  
        }  
    }  
}
```

- For API details, see [GetSecretValue](#) in *AWS SDK for Kotlin API reference*.

Python

## SDK for Python (Boto3)

## Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:
    """Encapsulates Secrets Manager functions."""
    def __init__(self, secretsmanager_client):
        """
        :param secretsmanager_client: A Boto3 Secrets Manager client.
        """
        self.secretsmanager_client = secretsmanager_client
        self.name = None

    def get_value(self, stage=None):
        """
        Gets the value of a secret.

        :param stage: The stage of the secret to retrieve. If this is None, the
                      current stage is retrieved.
        :return: The value of the secret. When the secret is a string, the value is
                 contained in the `SecretString` field. When the secret is bytes,
                 it is contained in the `SecretBinary` field.
        """
        if self.name is None:
            raise ValueError

        try:
            kwargs = {'SecretId': self.name}
            if stage is not None:
```

```
        kwargs['VersionStage'] = stage
    response = self.secretsmanager_client.get_secret_value(**kwargs)
    logger.info("Got value for secret %s.", self.name)
except ClientError:
    logger.exception("Couldn't get value for secret %s.", self.name)
    raise
else:
    return response
```

- For API details, see [GetSecretValue](#) in *AWS SDK for Python (Boto3) API Reference*.

Rust

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_secret(client: &Client, name: &str) -> Result<(), Error> {
    let resp = client.get_secret_value().secret_id(name).send().await?;

    println!("Value: {}", resp.secret_string().unwrap_or("No value!"));

    Ok(())
}
```

- For API details, see [GetSecretValue](#) in *AWS SDK for Rust API reference*.

## List Secrets Manager secrets using an AWS SDK

The following code examples show how to list Secrets Manager secrets.

Java

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllSecrets(SecretsManagerClient secretsClient) {
    try {
        ListSecretsResponse secretsResponse = secretsClient.listSecrets();
        List<SecretListEntry> secrets = secretsResponse.secretList();
        for (SecretListEntry secret: secrets) {
            System.out.println("The secret name is "+secret.name());
            System.out.println("The secret description is
"+secret.description());
        }
    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

- For API details, see [ListSecrets](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAllSecrets() {
    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        val response = secretsClient.listSecrets(ListSecretsRequest {})
        response.secretList?.forEach { secret ->
            println("The secret name is ${secret.name}")
            println("The secret description is ${secret.description}")
        }
    }
}
```

- For API details, see [ListSecrets](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:
    """Encapsulates Secrets Manager functions."""
    def __init__(self, secretsmanager_client):
        """
        :param secretsmanager_client: A Boto3 Secrets Manager client.
        """
        self.secretsmanager_client = secretsmanager_client
        self.name = None

    def list(self, max_results):
        """
        Lists secrets for the current account.

        :param max_results: The maximum number of results to return.
        :return: Yields secrets one at a time.
        """
        try:
            paginator = self.secretsmanager_client.get_paginator('list_secrets')
            for page in paginator.paginate(
                RegionName='us-east-1',
                PaginationConfig={}),
                    'Secrets': [
                        {
                            'Name': 'my-secret',
                            'Description': 'My AWS Lambda secret',
                            'Value': 'my-secret-value'
                        }
                    ]
            )
        except ClientError as e:
            print(f'An error occurred: {e}')
            raise e
        else:
            return page['Secrets']
```

```
PaginationConfig={'MaxItems': max_results}):
    for secret in page['SecretList']:
        yield secret
except ClientError:
    logger.exception("Couldn't list secrets.")
    raise
```

- For API details, see [ListSecrets](#) in *AWS SDK for Python (Boto3) API Reference*.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_secrets(client: &Client) -> Result<(), Error> {
    let resp = client.list_secrets().send().await?;

    println!("Secret names:");

    let secrets = resp.secret_list().unwrap_or_default();
    for secret in secrets {
        println!("  {}", secret.name().unwrap_or("No name!"));
    }

    println!("Found {} secrets", secrets.len());
    Ok(())
}
```

- For API details, see [ListSecrets](#) in *AWS SDK for Rust API reference*.

## Put a value in a Secrets Manager secret using an AWS SDK

The following code examples show how to put a value in a Secrets Manager secret.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void updateMySecret(SecretsManagerClient secretsClient, String
secretName, String secretValue) {

    try {
        UpdateSecretRequest secretRequest = UpdateSecretRequest.builder()
            .secretId(secretName)
            .secretString(secretValue)
```

```
        .build();

        secretsClient.updateSecret(secretRequest);

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutSecretValue](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun updateMySecret(secretName: String?, secretValue: String?) {

    val request = UpdateSecretRequest {
        secretId = secretName
        secretString = secretValue
    }

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        secretsClient.updateSecret(request)
        println("The secret value was updated")
    }
}
```

- For API details, see [PutSecretValue](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:
    """Encapsulates Secrets Manager functions."""
    def __init__(self, secretsmanager_client):
        """
        :param secretsmanager_client: A Boto3 Secrets Manager client.
        """
        self.secretsmanager_client = secretsmanager_client
        self.name = None

    def put_value(self, secret_value, stages=None):
        """
```

Puts a value into an existing secret. When no stages are specified, the value is set as the current ('AWSCURRENT') stage and the previous value is moved to the 'AWSPREVIOUS' stage. When a stage is specified that already exists, the stage is associated with the new value and removed from the old value.

```
:param secret_value: The value to add to the secret.
:param stages: The stages to associate with the secret.
:return: Metadata about the secret.
"""
if self.name is None:
    raise ValueError

try:
    kwargs = {'SecretId': self.name}
    if isinstance(secret_value, str):
        kwargs['SecretString'] = secret_value
    elif isinstance(secret_value, bytes):
        kwargs['SecretBinary'] = secret_value
    if stages is not None:
        kwargs['VersionStages'] = stages
    response = self.secretsmanager_client.put_secret_value(**kwargs)
    logger.info("Value put in secret %s.", self.name)
except ClientError:
    logger.exception("Couldn't put value in secret %s.", self.name)
    raise
else:
    return response
```

- For API details, see [PutSecretValue](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update the stage of a Secrets Manager secret version using an AWS SDK

The following code example shows how to update the stage of a Secrets Manager secret version.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:
    """Encapsulates Secrets Manager functions."""
    def __init__(self, secretsmanager_client):
        """
        :param secretsmanager_client: A Boto3 Secrets Manager client.
        """
        self.secretsmanager_client = secretsmanager_client
        self.name = None

    def update_version_stage(self, stage, remove_from, move_to):
        """
        Updates the stage associated with a version of the secret.

        :param stage: The stage to update.
        :param remove_from: The ID of the version to remove the stage from.
        :param move_to: The ID of the version to add the stage to.
        
```

```
:return: Metadata about the secret.  
"""  
    if self.name is None:  
        raise ValueError  
  
    try:  
        response = self.secretsmanager_client.update_secret_version_stage(  
            SecretId=self.name, VersionStage=stage,  
            RemoveFromVersionId=remove_from,  
            MoveToVersionId=move_to)  
        logger.info("Updated version stage %s for secret %s.", stage,  
self.name)  
    except ClientError:  
        logger.exception(  
            "Couldn't update version stage %s for secret %s.", stage,  
self.name)  
        raise  
    else:  
        return response
```

- For API details, see [UpdateSecretVersionStage](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios for Secrets Manager using AWS SDKs

The following code examples show how to use AWS Secrets Manager with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Create and manage a Secrets Manager secret using an AWS SDK \(p. 2140\)](#)

## Create and manage a Secrets Manager secret using an AWS SDK

The following code example shows how to:

- Create a Secrets Manager secret.
- Generate a random password and update a secret.
- Get current and previous values of a secret.
- Store binary data in a secret.
- Delete a secret.

### Python

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a secret, update its value and stage, and delete the secret.

```
def create_and_manage_secret_demo():  
    """  
        Shows how to use AWS Secrets Manager to create a secret, update its value and  
        stage, and delete it.  
    """  
    secret = SecretsManagerSecret(boto3.client('secretsmanager'))
```

```
print("Create a secret.")
secret.create("doc-example-secretsmanager-secret", "Shh, don't tell.")
print("Get secret value.")
value = secret.get_value()
print(f"Secret value: {value['SecretString']}")  
print("Get a random password.")
password = secret.get_random_password(20)
print(f"Got password: {password}")
print("Put password as new secret value.")
secret.put_value(password)
print("Get current and previous values.")
current = secret.get_value()
previous = secret.get_value('AWSVIOUS')
print(f"Current: {current['SecretString']}")  
print(f"Previous: {previous['SecretString']}")  
byteval = base64.b64encode("I'm a Base64 string!".encode('utf-8'))
stage = 'CUSTOM_STAGE'
print(f"Put byte value with a custom stage '{stage}'")
secret.put_value(byteval, [stage])
time.sleep(1)
print(f"Get secret value associated with stage '{stage}'")
got_val = secret.get_value(stage)
print(f"Raw bytes value: {got_val['SecretBinary']}")  
print(f"Decoded value:
{base64.b64decode(got_val['SecretBinary']).decode('utf-8')}")
pprint(secret.describe())
print("List 10 secrets for the account.")
for sec in secret.list(10):
    print(f"Name: {sec['Name']}")
print("Delete the secret.")
secret.delete(True)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateSecret](#)
  - [DeleteSecret](#)
  - [DescribeSecret](#)
  - [GetRandomPassword](#)
  - [GetSecretValue](#)
  - [ListSecrets](#)
  - [PutSecretValue](#)

## Cross-service examples for Secrets Manager using AWS SDKs

The following code examples show how to use AWS Secrets Manager with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Create a lending library REST API \(p. 2141\)](#)

## Create a lending library REST API

The following code example shows how to create a lending library where patrons can borrow and return books by using a REST API backed by an Amazon Aurora database.

## Python

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with the Amazon Relational Database Service (Amazon RDS) API and AWS Chalice to create a REST API backed by an Amazon Aurora database. The web service is fully serverless and represents a simple lending library where patrons can borrow and return books. Learn how to:

- Create and manage a serverless Aurora database cluster.
- Use AWS Secrets Manager to manage database credentials.
- Implement a data storage layer that uses Amazon RDS to move data into and out of the database.
- Use AWS Chalice to deploy a serverless REST API to Amazon API Gateway and AWS Lambda.
- Use the Requests package to send requests to the web service.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- API Gateway
- Lambda
- Amazon RDS
- Secrets Manager

## Code examples for Step Functions using AWS SDKs

The following code examples show how to use AWS Step Functions with an AWS software development kit (SDK).

### Code examples

- [Actions for Step Functions using AWS SDKs \(p. 2142\)](#)
  - Create a Step Functions state machine using an AWS SDK (p. 2143)
  - Delete a Step Functions state machine using an AWS SDK (p. 2145)
  - Describe a Step Functions state machine using an AWS SDK (p. 2147)
  - List Step Functions state machine runs using an AWS SDK (p. 2147)
  - List Step Functions state machines using an AWS SDK (p. 2149)
  - Start a Step Functions state machine run using an AWS SDK (p. 2151)
  - Stop a Step Functions state machine run using an AWS SDK (p. 2154)
  - Update a Step Functions state machine using an AWS SDK (p. 2154)
- [Cross-service examples for Step Functions using AWS SDKs \(p. 2155\)](#)
  - Create a messenger application with Step Functions (p. 2155)
  - Use Step Functions to invoke Lambda functions (p. 2156)

## Actions for Step Functions using AWS SDKs

The following code examples show how to use AWS Step Functions with AWS SDKs. Each example calls an individual service function.

### Examples

- [Create a Step Functions state machine using an AWS SDK \(p. 2143\)](#)

- [Delete a Step Functions state machine using an AWS SDK \(p. 2145\)](#)
- [Describe a Step Functions state machine using an AWS SDK \(p. 2147\)](#)
- [List Step Functions state machine runs using an AWS SDK \(p. 2147\)](#)
- [List Step Functions state machines using an AWS SDK \(p. 2149\)](#)
- [Start a Step Functions state machine run using an AWS SDK \(p. 2151\)](#)
- [Stop a Step Functions state machine run using an AWS SDK \(p. 2154\)](#)
- [Update a Step Functions state machine using an AWS SDK \(p. 2154\)](#)

## Create a Step Functions state machine using an AWS SDK

The following code examples show how to create a Step Functions state machine.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createMachine( SfnClient sfnClient, String roleARN, String
stateMachineName, String jsonFile) {

    String json = getJSONString(jsonFile);
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
            .definition(json)
            .name(stateMachineName)
            .roleArn(roleARN)
            .type(StateMachineType.STANDARD)
            .build();

        CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
        return response.stateMachineArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

private static String getJSONString(String path) {
    try {
        JSONParser parser = new JSONParser();
        JSONObject data = (JSONObject) parser.parse(new FileReader(path));// path to the JSON file.
        return data.toJSONString();

    } catch (IOException | org.json.simple.parser.ParseException e) {
        e.printStackTrace();
    }
    return "";
}
```

- For API details, see [CreateStateMachine](#) in [AWS SDK for Java 2.x API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createMachine( SfnClient sfnClient, String roleARN, String stateMachineName, String jsonFile ) {

    String json = getJSONString(jsonFile);
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
        .definition(json)
        .name(stateMachineName)
        .roleArn(roleARN)
        .type(StateMachineType.STANDARD)
        .build();

        CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
        return response.stateMachineArn();

    } catch ( SfnException e ) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

private static String getJSONString(String path) {
    try {
        JSONParser parser = new JSONParser();
        JSONObject data = (JSONObject) parser.parse(new FileReader(path));// path to the JSON file.
        return data.toJSONString();

    } catch ( IOException | org.json.simple.parser.ParseException e ) {
        e.printStackTrace();
    }
    return "";
}
```

- For API details, see [CreateStateMachine](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class StepFunctionsStateMachine:
```

```
"""Encapsulates Step Functions state machine functions."""
def __init__(self, stepfunctions_client):
    """
    :param stepfunctions_client: A Boto3 Step Functions client.
    """
    self.stepfunctions_client = stepfunctions_client
    self.state_machine_name = None
    self.state_machine_arn = None

def create(self, name, definition, role_arn):
    """
    Creates a new state machine.

    :param name: The name of the new state machine.
    :param definition: A dict that contains all of the state and flow control
        information. The dict is translated to JSON before it is uploaded.
    :param role_arn: A role that grants Step Functions permission to access any
        AWS services that are specified in the definition.
    :return: The Amazon Resource Name (ARN) of the new state machine.
    """
    try:
        response = self.stepfunctions_client.create_state_machine(
            name=name, definition=json.dumps(definition), roleArn=role_arn)
        self.state_machine_name = name
        self.state_machine_arn = response['stateMachineArn']
        logger.info(
            "Created state machine %s. ARN is %s.", name,
            self.state_machine_arn)
    except ClientError:
        logger.exception("Couldn't create state machine %s.", name)
        raise
    else:
        return self.state_machine_arn
```

- For API details, see [CreateStateMachine](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a Step Functions state machine using an AWS SDK

The following code examples show how to delete a Step Functions state machine.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {

    try {
        DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
        .stateMachineArn(stateMachineArn)
        .build();

        sfnClient.deleteStateMachine(deleteStateMachineRequest);
        System.out.println(stateMachineArn + " was successfully deleted.");

    } catch (SfnException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteStateMachine](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteMachine(stateMachineArnVal: String) {
    val deleteStateMachineRequest = DeleteStateMachineRequest {
        stateMachineArn = stateMachineArnVal
    }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        sfnClient.deleteStateMachine(deleteStateMachineRequest)
        println("$stateMachineArnVal was successfully deleted.")
    }
}
```

- For API details, see [DeleteStateMachine](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class StepFunctionsStateMachine:
    """Encapsulates Step Functions state machine functions."""
    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client
        self.state_machine_name = None
        self.state_machine_arn = None

    def delete(self):
        """
        Deletes a state machine and all associated run information.
        """
        if self.state_machine_arn is None:
            raise ValueError
        try:
            self.stepfunctions_client.delete_state_machine(

```

```
        stateMachineArn=self.state_machine_arn)
    logger.info("Deleted state machine %s.", self.state_machine_name)
    self._clear()
except ClientError:
    logger.exception(
        "Couldn't delete state machine %s.", self.state_machine_name)
    raise
```

- For API details, see [DeleteStateMachine](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a Step Functions state machine using an AWS SDK

The following code example shows how to describe a Step Functions state machine.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class StepFunctionsStateMachine:
    """Encapsulates Step Functions state machine functions."""
    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client
        self.state_machine_name = None
        self.state_machine_arn = None

    def describe(self):
        """
        Gets metadata about a state machine.

        :return: The metadata about the state machine.
        """
        if self.state_machine_arn is None:
            raise ValueError
        try:
            response = self.stepfunctions_client.describe_state_machine(
                stateMachineArn=self.state_machine_arn)
            logger.info("Got metadata for state machine %s.",
                        self.state_machine_name)
        except ClientError:
            logger.exception(
                "Couldn't get metadata for state machine %s.",
                self.state_machine_name)
            raise
        else:
            return response
```

- For API details, see [DescribeStateMachine](#) in *AWS SDK for Python (Boto3) API Reference*.

## List Step Functions state machine runs using an AWS SDK

The following code examples show how to list Step Functions state machine runs.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getExeHistory(SfnClient sfnClient, String exeARN) {
    try {
        GetExecutionHistoryRequest historyRequest =
GetExecutionHistoryRequest.builder()
            .executionArn(exeARN)
            .maxResults(10)
            .build();

        GetExecutionHistoryResponse historyResponse =
sfnClient.getExecutionHistory(historyRequest);
        List<HistoryEvent> events = historyResponse.events();
        for (HistoryEvent event: events) {
            System.out.println("The event type is "+event.type().toString());
        }

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListExecutions](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getExeHistory(exeARN: String?) {

    val historyRequest = GetExecutionHistoryRequest {
        executionArn = exeARN
        maxResults = 10
    }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getExecutionHistory(historyRequest)
        response.events?.forEach { event ->
            println("The event type is ${event.type}")
        }
    }
}
```

- For API details, see [ListExecutions](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class StepFunctionsStateMachine:  
    """Encapsulates Step Functions state machine functions."""  
    def __init__(self, stepfunctions_client):  
        """  
        :param stepfunctions_client: A Boto3 Step Functions client.  
        """  
        self.stepfunctions_client = stepfunctions_client  
        self.state_machine_name = None  
        self.state_machine_arn = None  
  
    def list_runs(self, run_status=None):  
        """  
        Lists the runs for the state machine.  
  
        :param run_status: When specified, only lists runs that have the specified  
                          status. Otherwise, all runs are listed.  
        :return: The list of runs.  
        """  
        if self.state_machine_arn is None:  
            raise ValueError  
        try:  
            kwargs = {'stateMachineArn': self.state_machine_arn}  
            if run_status is not None:  
                kwargs['statusFilter'] = run_status  
            response = self.stepfunctions_client.list_executions(**kwargs)  
            runs = response['executions']  
            logger.info(  
                "Got %s runs for state machine %s.", len(runs),  
                self.state_machine_name)  
        except ClientError:  
            logger.exception(  
                "Couldn't get runs for state machine %s.", self.state_machine_name)  
            raise  
        else:  
            return runs
```

- For API details, see [ListExecutions](#) in *AWS SDK for Python (Boto3) API Reference*.

## List Step Functions state machines using an AWS SDK

The following code examples show how to list Step Functions state machines.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listMachines(SfnClient sfnClient) {  
  
    try {  
        ListStateMachinesResponse response = sfnClient.listStateMachines();  
        List<StateMachineListItem> machines = response.stateMachines();  
        for (StateMachineListItem machine :machines) {  
            System.out.println("The name of the state machine is:  
"+machine.name());  
            System.out.println("The ARN value is :  
"+machine.stateMachineArn());  
        }  
  
    } catch (SfnException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListStateMachines](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listMachines() {  
  
    SfnClient { region = "us-east-1" }.use { sfnClient ->  
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})  
        response.stateMachines?.forEach { machine ->  
            println("The name of the state machine is ${machine.name}")  
            println("The ARN value is ${machine.stateMachineArn}")  
        }  
    }  
}
```

- For API details, see [ListStateMachines](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Find a state machine by name by searching the list of state machines for the account.

```
class StepFunctionsStateMachine:
```

```
"""Encapsulates Step Functions state machine functions."""
def __init__(self, stepfunctions_client):
    """
    :param stepfunctions_client: A Boto3 Step Functions client.
    """
    self.stepfunctions_client = stepfunctions_client
    self.state_machine_name = None
    self.state_machine_arn = None

    def find(self, state_machine_name):
        """
        Finds a state machine by name. This function iterates the state machines
        for the current account until it finds a match and returns the first matching
        state machine.

        :param state_machine_name: The name of the state machine to find.
        :return: The ARN of the named state machine when found; otherwise, None.
        """
        self._clear()
        try:
            paginator =
                self.stepfunctions_client.getPaginator('list_state_machines')
            for page in paginator.paginate():
                for machine in page['stateMachines']:
                    if machine['name'] == state_machine_name:
                        self.state_machine_name = state_machine_name
                        self.state_machine_arn = machine['stateMachineArn']
                        break
                    if self.state_machine_arn is not None:
                        break
            if self.state_machine_arn is not None:
                logger.info(
                    "Found state machine %s with ARN %s.", self.state_machine_name,
                    self.state_machine_arn)
            else:
                logger.info("Couldn't find state machine %s.", state_machine_name)
        except ClientError:
            logger.exception("Couldn't find state machine %s.", state_machine_name)
            raise
        else:
            return self.state_machine_arn
```

- For API details, see [ListStateMachines](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start a Step Functions state machine run using an AWS SDK

The following code examples show how to start a Step Functions state machine run.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonFile) {

    String json = getJSONString(jsonFile);
```

```
UUID uuid = UUID.randomUUID();
String uuidValue = uuid.toString();
try {

    StartExecutionRequest executionRequest =
StartExecutionRequest.builder()
    .input(json)
    .stateMachineArn(stateMachineArn)
    .name(uuidValue)
    .build();

    StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
    return response.executionArn();

} catch (SfnException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

private static String getJSONString(String path) {

try {
    JSONParser parser = new JSONParser();
    JSONObject data = (JSONObject) parser.parse(new FileReader(path));// path to the JSON file.
    String json = data.toJSONString();
    return json;

} catch (IOException | org.json.simple.parser.ParseException e) {
    e.printStackTrace();
}
return "";
}
```

- For API details, see [StartExecution](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun startWorkflow(stateMachineArnVal: String?, jsonFile: String): String? {
    val json = getJSONString(jsonFile)

    // Specify the name of the execution by using a GUID value.
    val uuid = UUID.randomUUID()
    val uuidValue = uuid.toString()
    val request = StartExecutionRequest {
        input = json
        stateMachineArn = stateMachineArnVal
        name = uuidValue
    }
```

```
        SfnClient { region = "us-east-1" }.use { sfnClient ->
            val response = sfnClient.startExecution(request)
            return response.executionArn
        }
    }

private fun getJSONString(path: String): String {
    try {
        val parser = JSONParser()
        val data = parser.parse(FileReader(path)) as JSONObject // path to the JSON
        file.
        return data.toJSONString()
    } catch (e: IOException) {
        print(e.message)
    } catch (e: ParseException) {
        print(e.message)
    }
    return ""
}
```

- For API details, see [StartExecution](#) in *AWS SDK for Kotlin API reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class StepFunctionsStateMachine:
    """Encapsulates Step Functions state machine functions."""
    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client
        self.state_machine_name = None
        self.state_machine_arn = None

    def start_run(self, run_name, run_input=None):
        """
        Starts a run with the current state definition.

        :param run_name: The name of the run. This name must be unique for all runs
                         for the state machine.
        :param run_input: Data that is passed as input to the run.
        :return: The ARN of the run.
        """
        if self.state_machine_arn is None:
            raise ValueError
        try:
            kwargs = {'stateMachineArn': self.state_machine_arn, 'name': run_name}
            if run_input is not None:
                kwargs['input'] = json.dumps(run_input)
            response = self.stepfunctions_client.start_execution(**kwargs)
            run_arn = response['executionArn']
            logger.info("Started run %s. ARN is %s.", run_name, run_arn)
        except ClientError:
            logger.exception("Couldn't start run %s.", run_name)
```

```
        raise
    else:
        return run_arn
```

- For API details, see [StartExecution](#) in *AWS SDK for Python (Boto3) API Reference*.

## Stop a Step Functions state machine run using an AWS SDK

The following code example shows how to stop a Step Functions state machine run.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class StepFunctionsStateMachine:
    """Encapsulates Step Functions state machine functions."""
    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client
        self.state_machine_name = None
        self.state_machine_arn = None

    def stop_run(self, run_arn, cause):
        """
        Stops a run.

        :param run_arn: The run to stop.
        :param cause: A description of why the run was stopped.
        """
        try:
            self.stepfunctions_client.stop_execution(executionArn=run_arn,
                                                      cause=cause)
            logger.info("Stopping run %s.", run_arn)
        except ClientError:
            logger.exception("Couldn't stop run %s.", run_arn)
            raise
```

- For API details, see [StopExecution](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update a Step Functions state machine using an AWS SDK

The following code example shows how to update a Step Functions state machine.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class StepFunctionsStateMachine:  
    """Encapsulates Step Functions state machine functions."""  
    def __init__(self, stepfunctions_client):  
        """  
        :param stepfunctions_client: A Boto3 Step Functions client.  
        """  
        self.stepfunctions_client = stepfunctions_client  
        self.state_machine_name = None  
        self.state_machine_arn = None  
  
    def update(self, definition, role_arn=None):  
        """  
        Updates an existing state machine. Any runs currently operating do not  
        update until they are stopped.  
  
        :param definition: A dict that contains all of the state and flow control  
                           information for the state machine. This completely  
                           replaces  
                           the existing definition.  
        :param role_arn: A role that grants Step Functions permission to access any  
                         AWS services that are specified in the definition.  
        """  
        if self.state_machine_arn is None:  
            raise ValueError  
        try:  
            kwargs = {  
                'stateMachineArn': self.state_machine_arn,  
                'definition': json.dumps(definition)}  
            if role_arn is not None:  
                kwargs['roleArn'] = role_arn  
            self.stepfunctions_client.update_state_machine(**kwargs)  
            logger.info("Updated state machine %s.", self.state_machine_name)  
        except ClientError:  
            logger.exception(  
                "Couldn't update state machine %s.", self.state_machine_name)  
            raise
```

- For API details, see [UpdateStateMachine](#) in *AWS SDK for Python (Boto3) API Reference*.

## Cross-service examples for Step Functions using AWS SDKs

The following code examples show how to use AWS Step Functions with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Create a messenger application with Step Functions \(p. 2155\)](#)
- [Use Step Functions to invoke Lambda functions \(p. 2156\)](#)

## Create a messenger application with Step Functions

The following code example shows how to create an AWS Step Functions messenger application that retrieves message records from a database table.

## Python

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with AWS Step Functions to create a messenger application that retrieves message records from an Amazon DynamoDB table and sends them with Amazon Simple Queue Service (Amazon SQS). The state machine integrates with an AWS Lambda function to scan the database for unsent messages.

- Create a state machine that retrieves and updates message records from an Amazon DynamoDB table.
- Update the state machine definition to also send messages to Amazon Simple Queue Service (Amazon SQS).
- Start and stop state machine runs.
- Connect to Lambda, DynamoDB, and Amazon SQS from a state machine by using service integrations.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Lambda
- Amazon SQS
- Step Functions

## Use Step Functions to invoke Lambda functions

The following code examples show how to create an AWS Step Functions state machine that invokes AWS Lambda functions in sequence.

### Java

#### SDK for Java 2.x

Shows how to create an AWS serverless workflow by using AWS Step Functions and the AWS SDK for Java 2.x. Each workflow step is implemented using an AWS Lambda function.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

### JavaScript

#### SDK for JavaScript V3

Shows how to create an AWS serverless workflow by using AWS Step Functions and the AWS SDK for JavaScript. Each workflow step is implemented using an AWS Lambda function.

Lambda is a compute service that enables you to run code without provisioning or managing servers. Step Functions is a serverless orchestration service that lets you combine Lambda functions and other AWS services to build business-critical applications.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

### Services used in this example

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

## Code examples for AWS Support using AWS SDKs

The following code examples show how to use AWS Support with an AWS software development kit (SDK).

### Get started

#### Hello AWS Support

The following code examples show how to get started using AWS Support.

##### .NET

###### AWS SDK for .NET

###### Note

There's more on [GitHub](#). Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon AWSSupport;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

public static class HelloSupport
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
        // the AWS Support service.
        // Use your AWS profile name, or leave it blank to use the default profile.
        // You must have one of the following AWS Support plans: Business,
        Enterprise On-Ramp, or Enterprise. Otherwise, an exception will be thrown.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices(_ , services) =>
                services.AddAWSService<IAmazonAWSSupport>()
            .Build();

        // Now the client is available for injection.
        var supportClient = host.Services.GetRequiredService<IAmazonAWSSupport>();
```

```
// You can use await and any of the async methods to get a response.  
var response = await supportClient.DescribeServicesAsync();  
Console.WriteLine($"\\tHello AWS Support! There are  
{response.Services.Count} services available.");  
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
 * Before running this Java (v2) code example, set up your development environment,  
 * including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * In addition, you must have the AWS Business Support Plan to use the AWS Support  
 * Java API. For more information, see:  
 *  
 * https://aws.amazon.com/premiumsupport/plans/  
 *  
 * This Java example performs the following task:  
 *  
 * 1. Gets and displays available services.  
 *  
 *  
 * NOTE: To see multiple operations, see SupportScenario.  
 */  
  
public class HelloSupport {  
  
    public static void main(String[] args) {  
        Region region = Region.US_WEST_2;  
        SupportClient supportClient = SupportClient.builder()  
            .region(region)  
            .build();  
  
        System.out.println("***** Step 1. Get and display available services.");  
        displayServices(supportClient);  
    }  
  
    // Return a List that contains a Service name and Category name.  
    public static void displayServices(SupportClient supportClient) {  
        try {  
            DescribeServicesRequest servicesRequest =  
                DescribeServicesRequest.builder()  
                    .language("en")  
                    .build();  
  
            DescribeServicesResponse response =  
                supportClient.describeServices(servicesRequest);  
        }  
    }  
}
```

```
List<Service> services = response.services();

System.out.println("Get the first 10 services");
int index = 1;
for (Service service: services) {
    if (index== 11)
        break;

    System.out.println("The Service name is: "+service.name());

    // Display the Categories for this service.
    List<Category> categories = service.categories();
    for (Category cat: categories) {
        System.out.println("The category name is: "+cat.name());
    }
    index++ ;
}

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/** 
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

In addition, you must have the AWS Business Support Plan to use the AWS Support
Java API. For more information, see:

https://aws.amazon.com/premiumsupport/plans/

This Kotlin example performs the following task:

1. Gets and displays available services.
 */

suspend fun main() {
    displaySomeServices()
}

// Return a List that contains a Service name and Category name.
```

```
suspend fun displaySomeServices() {
    val servicesRequest = DescribeServicesRequest {
        language = "en"
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is: " + service.name)

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                index++
            }
        }
    }
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for Kotlin API reference*.

## Code examples

- [Actions for AWS Support using AWS SDKs \(p. 2160\)](#)
  - [Add an AWS Support communication to a case using an AWS SDK \(p. 2161\)](#)
  - [Add an AWS Support attachment to a set using an AWS SDK \(p. 2163\)](#)
  - [Create an AWS Support case using an AWS SDK \(p. 2164\)](#)
  - [Describe an attachment for an AWS Support case using an AWS SDK \(p. 2167\)](#)
  - [Describe AWS Support cases using an AWS SDK \(p. 2168\)](#)
  - [Describe AWS Support communications for a case using an AWS SDK \(p. 2170\)](#)
  - [Describe the available AWS services for support cases using an AWS SDK \(p. 2172\)](#)
  - [Describe AWS Support severity levels using an AWS SDK \(p. 2175\)](#)
  - [Resolve an AWS Support case using an AWS SDK \(p. 2176\)](#)
- [Scenarios for AWS Support using AWS SDKs \(p. 2178\)](#)
  - [Get started with AWS Support cases using an AWS SDK \(p. 2178\)](#)

## Actions for AWS Support using AWS SDKs

The following code examples show how to use AWS Support with AWS SDKs. Each example calls an individual service function.

### Examples

- [Add an AWS Support communication to a case using an AWS SDK \(p. 2161\)](#)
- [Add an AWS Support attachment to a set using an AWS SDK \(p. 2163\)](#)
- [Create an AWS Support case using an AWS SDK \(p. 2164\)](#)
- [Describe an attachment for an AWS Support case using an AWS SDK \(p. 2167\)](#)

- [Describe AWS Support cases using an AWS SDK \(p. 2168\)](#)
- [Describe AWS Support communications for a case using an AWS SDK \(p. 2170\)](#)
- [Describe the available AWS services for support cases using an AWS SDK \(p. 2172\)](#)
- [Describe AWS Support severity levels using an AWS SDK \(p. 2175\)](#)
- [Resolve an AWS Support case using an AWS SDK \(p. 2176\)](#)

## Add an AWS Support communication to a case using an AWS SDK

The following code examples show how to add an AWS Support communication with an attachment to a support case.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Add communication to a case, including optional attachment set ID and CC
email addresses.
///</summary>
///<param name="caseId">Id for the support case.</param>
///<param name="body">Body text of the communication.</param>
///<param name="attachmentSetId">Optional Id for an attachment set.</param>
///<param name="ccEmailAddresses">Optional list of CC email addresses.</param>
///<returns>True if successful.</returns>
public async Task<bool> AddCommunicationToCase(string caseId, string body,
    string? attachmentSetId = null, List<string>? ccEmailAddresses = null)
{
    var response = await _amazonSupport.AddCommunicationToCaseAsync(
        new AddCommunicationToCaseRequest()
    {
        CaseId = caseId,
        CommunicationBody = body,
        AttachmentSetId = attachmentSetId,
        CcEmailAddresses = ccEmailAddresses
    });
    return response.Result;
}
```

- For API details, see [AddCommunicationToCase](#) in *AWS SDK for .NET API Reference*.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
        .caseId(caseId)
        .attachmentSetId(attachmentSetId)
        .communicationBody("Please refer to attachment for details.")
        .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
        else
            System.out.println("There was an error adding the communication to
an AWS Support case");

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [AddCommunicationToCase](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun addAttachSupportCase(caseIdVal: String?, attachmentSetIdVal: String?) {
    val caseRequest = AddCommunicationToCaseRequest {
        caseId = caseIdVal
        attachmentSetId = attachmentSetIdVal
        communicationBody = "Please refer to attachment for details."
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result()) {
            println("You have successfully added a communication to an AWS Support
case")
        } else {
            println("There was an error adding the communication to an AWS Support
case")
        }
    }
}
```

- For API details, see [AddCommunicationToCase](#) in *AWS SDK for Kotlin API reference*.

## Add an AWS Support attachment to a set using an AWS SDK

The following code examples show how to add an AWS Support attachment to an attachment set.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Add an attachment to a set, or create a new attachment set if one does not
exist.
/// </summary>
/// <param name="data">The data for the attachment.</param>
/// <param name="fileName">The file name for the attachment.</param>
/// <param name="attachmentSetId">Optional setId for the attachment. Creates a
new attachment set if empty.</param>
/// <returns>The setId of the attachment.</returns>
public async Task<string> AddAttachmentToSet(MemoryStream data, string
fileName, string? attachmentSetId = null)
{
    var response = await _amazonSupport.AddAttachmentsToSetAsync(
        new AddAttachmentsToSetRequest
    {
        AttachmentSetId = attachmentSetId,
        Attachments = new List<Attachment>
        {
            new Attachment
            {
                Data = data,
                FileName = fileName
            }
        }
    });
    return response.AttachmentSetId;
}
```

- For API details, see [AddAttachmentsToSet](#) in *AWS SDK for .NET API Reference*.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
```

```
Attachment attachment = Attachment.builder()
    .fileName(myFile.getName())
    .data(sourceBytes)
    .build();

AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
    .attachments(attachment)
    .build();

AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
return response.attachmentSetId();

} catch (SupportException | FileNotFoundException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [AddAttachmentsToSet](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal = Attachment {
        fileName = myFile.name
        data = sourceBytes
    }

    val setRequest = AddAttachmentsToSetRequest {
        attachments = listOf(attachmentVal)
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}
```

- For API details, see [AddAttachmentsToSet](#) in *AWS SDK for Kotlin API reference*.

## Create an AWS Support case using an AWS SDK

The following code examples show how to create a new AWS Support case.

.NET

**AWS SDK for .NET**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create a new support case.
/// </summary>
/// <param name="serviceCode">Service code for the new case.</param>
/// <param name="categoryCode">Category for the new case.</param>
/// <param name="severityCode">Severity code for the new case.</param>
/// <param name="subject">Subject of the new case.</param>
/// <param name="body">Body text of the new case.</param>
/// <param name="language">Optional language support for your case.
/// Currently "en" (English) and "ja" (Japanese) are supported.</param>
/// <param name="attachmentSetId">Optional Id for an attachment set for the new
case.</param>
    /// <param name="issueType">Optional issue type for the new case. Options are
"customer-service" or "technical".</param>
    /// <returns>The caseId of the new support case.</returns>
    public async Task<string> CreateCase(string serviceCode, string categoryCode,
string severityCode, string subject,
        string body, string language = "en", string? attachmentSetId = null, string
issueType = "customer-service")
    {
        var response = await _amazonSupport.CreateCaseAsync(
            new CreateCaseRequest()
            {
                ServiceCode = serviceCode,
                CategoryCode = categoryCode,
                SeverityCode = severityCode,
                Subject = subject,
                Language = language,
                AttachmentSetId = attachmentSetId,
                IssueType = issueType,
                CommunicationBody = body
            });
        return response.CaseId;
    }
}
```

- For API details, see [CreateCase in AWS SDK for .NET API Reference](#).

Java

**SDK for Java 2.x**

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createSupportCase(SupportClient supportClient,
List<String> sevCatList, String sevLevel) {
    try {
        String serviceCode = sevCatList.get(0);
```

```
String caseCat = sevCatList.get(1);
CreateCaseRequest caseRequest = CreateCaseRequest.builder()
    .categoryCode(caseCat.toLowerCase())
    .serviceCode(serviceCode.toLowerCase())
    .severityCode(sevLevel.toLowerCase())
    .communicationBody("Test issue with "+serviceCode.toLowerCase())
    .subject("Test case, please ignore")
    .language("en")
    .issueType("technical")
    .build();

CreateCaseResponse response = supportClient.createCase(caseRequest);
return response.caseId();

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [CreateCase in AWS SDK for Java 2.x API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createSupportCase(sevCatListVal: List<String>, sevLevelVal: String): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest = CreateCaseRequest {
        categoryCode = caseCategory.lowercase(Locale.getDefault())
        serviceCode = serCode.lowercase(Locale.getDefault())
        severityCode = sevLevelVal.lowercase(Locale.getDefault())
        communicationBody = "Test issue with
${serCode.lowercase(Locale.getDefault())}"
        subject = "Test case, please ignore"
        language = "en"
        issueType = "technical"
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}
```

- For API details, see [CreateCase in AWS SDK for Kotlin API reference](#).

## Describe an attachment for an AWS Support case using an AWS SDK

The following code examples show how to describe an attachment for an AWS Support case.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get description of a specific attachment.
/// </summary>
/// <param name="attachmentId">Id of the attachment, usually fetched by
describing the communications of a case.</param>
/// <returns>The attachment object.</returns>
public async Task<Attachment> DescribeAttachment(string attachmentId)
{
    var response = await _amazonSupport.DescribeAttachmentAsync(
        new DescribeAttachmentRequest()
    {
        AttachmentId = attachmentId
    });
    return response.Attachment;
}
```

- For API details, see [DescribeAttachment](#) in [AWS SDK for .NET API Reference](#).

### Java

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeAttachment(SupportClient supportClient, String
attachId) {
    try {
        DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
            .attachmentId(attachId)
            .build();

        DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
        System.out.println("The name of the file is
"+response.attachment().fileName());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeAttachment](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeAttachment(attachId: String?) {  
    val attachmentRequest = DescribeAttachmentRequest {  
        attachmentId = attachId  
    }  
  
    SupportClient { region = "us-west-2" }.use { supportClient ->  
        val response = supportClient.describeAttachment(attachmentRequest)  
        println("The name of the file is ${response.attachment?.fileName}")  
    }  
}
```

- For API details, see [DescribeAttachment](#) in *AWS SDK for Kotlin API reference*.

## Describe AWS Support cases using an AWS SDK

The following code examples show how to describe AWS Support cases.

## .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Get case details for a list of case ids, optionally with date filters.  
/// </summary>  
/// <param name="caseIds">The list of case IDs.</param>  
/// <param name="displayId">Optional display ID.</param>  
/// <param name="includeCommunication">True to include communication. Defaults to true.</param>  
/// <param name="includeResolvedCases">True to include resolved cases. Defaults to false.</param>  
/// <param name="afterTime">The optional start date for a filtered search.</param>  
/// <param name="beforeTime">The optional end date for a filtered search.</param>  
/// <param name="language">Optional language support for your case.  
/// Currently "en" (English) and "ja" (Japanese) are supported.</param>  
/// <returns>A list of CaseDetails.</returns>
```

```
public async Task<List<CaseDetails>> DescribeCases(List<string> caseIds,
    string? displayId = null, bool includeCommunication = true,
    bool includeResolvedCases = false, DateTime? afterTime = null, DateTime?
    beforeTime = null,
    string language = "en")
{
    var results = new List<CaseDetails>();
    var paginateCases = _amazonSupport.Paginator.DescribeCases(
        new DescribeCasesRequest()
    {
        CaseIdList = caseIds,
        DisplayId = displayId,
        IncludeCommunications = includeCommunication,
        IncludeResolvedCases = includeResolvedCases,
        AfterTime = afterTime?.ToString("o"),
        BeforeTime = beforeTime?.ToString("o"),
        Language = language
    });
    // Get the entire list using the paginator.
    await foreach (var cases in paginateCases.Cases)
    {
        results.Add(cases);
    }
    return results;
}
```

- For API details, see [DescribeCases](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
        DescribeCasesRequest.builder()
            .maxResults(20)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
            .build();

        DescribeCasesResponse response =
        supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase: cases) {
            System.out.println("The case status is "+sinCase.status());
            System.out.println("The case Id is "+sinCase.caseId());
            System.out.println("The case subject is "+sinCase.subject());
        }
    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
    }
}
```

```
        System.exit(1);
    }
```

- For API details, see [DescribeCases](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest = DescribeCasesRequest {
        maxResults = 20
        afterTime = yesterday.toString()
        beforeTime = now.toString()
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}
```

- For API details, see [DescribeCases](#) in *AWS SDK for Kotlin API reference*.

## Describe AWS Support communications for a case using an AWS SDK

The following code examples show how to describe AWS Support communications for a case.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Describe the communications for a case, optionally with a date filter.

```

```
    /// </summary>
    /// <param name="caseId">The ID of the support case.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</param>
    /// <param name="beforeTime">The optional end date for a filtered search.</param>
    /// <returns>The list of communications for the case.</returns>
    public async Task<List<Communication>> DescribeCommunications(string caseId,
    DateTime? afterTime = null, DateTime? beforeTime = null)
    {
        var results = new List<Communication>();
        var paginateCommunications =
        _amazonSupport.Paginator.DescribeCommunications(
            new DescribeCommunicationsRequest()
            {
                CaseId = caseId,
                AfterTime = afterTime?.ToString("G"),
                BeforeTime = beforeTime?.ToString("G")
            });
        // Get the entire list using the paginator.
        await foreach (var communications in paginateCommunications.Communications)
        {
            results.Add(communications);
        }
        return results;
    }
```

- For API details, see [DescribeCommunications](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String listCommunications(SupportClient supportClient, String
caseId) {
    try {
        String attachId = null;
        DescribeCommunicationsRequest communicationsRequest =
        DescribeCommunicationsRequest.builder()
            .caseId(caseId)
            .maxResults(10)
            .build();

        DescribeCommunicationsResponse response =
        supportClient.describeCommunications(communicationsRequest);
        List<Communication> communications = response.communications();
        for (Communication comm: communications) {
            System.out.println("the body is: " + comm.body());

            //Get the attachment id value.
            List<AttachmentDetails> attachments = comm.attachmentSet();
            for (AttachmentDetails detail : attachments) {
                attachId = detail.attachmentId();
            }
        }
    return attachId;
}
```

```
        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }
```

- For API details, see [DescribeCommunications](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest = DescribeCommunicationsRequest {
        caseId = caseIdVal
        maxResults = 10
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}
```

- For API details, see [DescribeCommunications](#) in *AWS SDK for Kotlin API reference*.

## Describe the available AWS services for support cases using an AWS SDK

The following code examples show how to describe the list of AWS services.

### .NET

#### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get the descriptions of AWS services.

```

```
/// </summary>
/// <param name="name">Optional language for services.
/// Currently "en" (English) and "ja" (Japanese) are supported.</param>
/// <returns>The list of AWS service descriptions.</returns>
public async Task<List<Service>> DescribeServices(string language = "en")
{
    var response = await _amazonSupport.DescribeServicesAsync(
        new DescribeServicesRequest()
    {
        Language = language
    });
    return response.Services;
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
    .language("en")
    .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        String serviceCode = null;
        String catName = null;
        List<String> sevCatList = new ArrayList<>();
        List<Service> services = response.services();

        System.out.println("Get the first 10 services");
        int index = 1;
        for (Service service: services) {
            if (index== 11)
                break;

            System.out.println("The Service name is: "+service.name());
            if (service.name().compareTo("Account") == 0)
                serviceCode = service.code();

            // Get the Categories for this service.
            List<Category> categories = service.categories();
            for (Category cat: categories) {
                System.out.println("The category name is: "+cat.name());
                if (cat.name().compareTo("Security") == 0)
                    catName = cat.name();
            }
            index++ ;
        }

        // Push the two values to the list.
        sevCatList.add(serviceCode);
```

```
        sevCatList.add(catName);
        return sevCatList;

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return null;
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest = DescribeServicesRequest {
        language = "en"
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }
        }

        // Get the categories for this service.
        service.categories?.forEach { cat ->
            println("The category name is ${cat.name}")
            if (cat.name == "Security") {
                catName = cat.name!!
            }
        }
        index++
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
}
```

```
        return sevCatList
    }
```

- For API details, see [DescribeServices](#) in *AWS SDK for Kotlin API reference*.

## Describe AWS Support severity levels using an AWS SDK

The following code examples show how to describe AWS Support severity levels.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get the descriptions of support severity levels.
/// </summary>
/// <param name="name">Optional language for severity levels.
/// Currently "en" (English) and "ja" (Japanese) are supported.</param>
/// <returns>The list of support severity levels.</returns>
public async Task<List<SeverityLevel>> DescribeSeverityLevels(string language =
"en")
{
    var response = await _amazonSupport.DescribeSeverityLevelsAsync(
        new DescribeSeverityLevelsRequest()
    {
        Language = language
    });
    return response.SeverityLevels;
}
```

- For API details, see [DescribeSeverityLevels](#) in *AWS SDK for .NET API Reference*.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String displaySevLevels(SupportClient supportClient) {
    try {
        DescribeSeverityLevelsRequest severityLevelsRequest =
            DescribeSeverityLevelsRequest.builder()
                .language("en")
                .build();

        DescribeSeverityLevelsResponse response =
            supportClient.describeSeverityLevels(severityLevelsRequest);
        List<SeverityLevel> severityLevels = response.severityLevels();
        String levelName = null;
        for (SeverityLevel sevLevel: severityLevels) {
```

```
        System.out.println("The severity level name is: "+  
        sevLevel.name());  
        if (sevLevel.name().compareTo("High")==0)  
            levelName = sevLevel.name();  
    }  
    return levelName;  
  
} catch (SupportException e) {  
    System.out.println(e.getLocalizedMessage());  
    System.exit(1);  
}  
return "";  
}
```

- For API details, see [DescribeSeverityLevels](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun displaySevLevels(): String {  
    var levelName = ""  
    val severityLevelsRequest = DescribeSeverityLevelsRequest {  
        language = "en"  
    }  
  
    SupportClient { region = "us-west-2" }.use { supportClient ->  
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)  
        response.severityLevels?.forEach { sevLevel ->  
            println("The severity level name is: ${sevLevel.name}")  
            if (sevLevel.name == "High") {  
                levelName = sevLevel.name!!  
            }  
        }  
        return levelName  
    }  
}
```

- For API details, see [DescribeSeverityLevels](#) in *AWS SDK for Kotlin API reference*.

## Resolve an AWS Support case using an AWS SDK

The following code examples show how to resolve an AWS Support case.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Resolve a support case by caseId.
/// </summary>
/// <param name="caseId">Id for the support case.</param>
/// <returns>The final status of the case after resolving.</returns>
public async Task<string> ResolveCase(string caseId)
{
    var response = await _amazonSupport.ResolveCaseAsync(
        new ResolveCaseRequest()
    {
        CaseId = caseId
    });
    return response.FinalCaseStatus;
}
```

- For API details, see [ResolveCase in AWS SDK for .NET API Reference](#).

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case "+caseId+" is
"+response.finalCaseStatus());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [ResolveCase in AWS SDK for Java 2.x API Reference](#).

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest = ResolveCaseRequest {
        caseId = caseIdVal
    }
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}
```

- For API details, see [ResolveCase in AWS SDK for Kotlin API reference](#).

## Scenarios for AWS Support using AWS SDKs

The following code examples show how to use AWS Support with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Get started with AWS Support cases using an AWS SDK \(p. 2178\)](#)

## Get started with AWS Support cases using an AWS SDK

The following code examples show how to:

- Get and display available services and severity levels for cases.
- Create a support case using a selected service, category, and severity level.
- Get and display a list of open cases for the current day.
- Add an attachment set and a communication to the new case.
- Describe the new attachment and communication for the case.
- Resolve the case.
- Get and display a list of resolved cases for the current day.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
/// <summary>
/// Hello AWS Support example.
/// </summary>
public static class SupportCaseScenario
{
    /*
     Before running this .NET code example, set up your development environment,
     including your credentials.
     To use the AWS Support API, you must have one of the following AWS Support
     plans: Business, Enterprise On-Ramp, or Enterprise.

    This .NET example performs the following tasks:
```

```
1. Get and display services. Select a service from the list.
2. Select a category from the selected service.
3. Get and display severity levels and select a severity level from the list.
4. Create a support case using the selected service, category, and severity
level.
5. Get and display a list of open support cases for the current day.
6. Create an attachment set with a sample text file to add to the case.
7. Add a communication with the attachment to the support case.
8. List the communications of the support case.
9. Describe the attachment set.
10. Resolve the support case.
11. Get a list of resolved cases for the current day.
*/
private static SupportWrapper _supportWrapper = null!;

static async Task Main(string[] args)
{
    // Set up dependency injection for the AWS Support service.
    // Use your AWS profile name, or leave it blank to use the default profile.
    using var host = Host.CreateDefaultBuilder(args)
        .ConfigureLogging(logging =>
            logging.AddFilter("System", LogLevel.Debug)
                .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
        .ConfigureServices(_>, services =>
            services.AddAWSService<IAmazonAWSSupport>(new AWSOptions()
{ Profile = "default" })
                .AddTransient<SupportWrapper>()
        )
        .Build();

    var logger = LoggerFactory.Create(builder =>
    {
        builder.AddConsole();
    }).CreateLogger(typeof(SupportCaseScenario));

    _supportWrapper = host.Services.GetRequiredService<SupportWrapper>();

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Welcome to the AWS Support case example scenario.");
    Console.WriteLine(new string('-', 80));

    try
    {
        var apiSupported = await _supportWrapper.VerifySubscription();
        if (!apiSupported)
        {
            logger.LogError("You must have a Business, Enterprise On-Ramp, or
Enterprise Support " +
                            "plan to use the AWS Support API. \n\tPlease
upgrade your subscription to run these examples.");
            return;
        }

        var service = await DisplayAndSelectServices();

        var category = DisplayAndSelectCategories(service);

        var severityLevel = await DisplayAndSelectSeverity();

        var caseId = await CreateSupportCase(service, category, severityLevel);

        await DescribeTodayOpenCases();
    }
}
```

```
        var attachmentSetId = await CreateAttachmentSet();

        await AddCommunicationToCase(attachmentSetId, caseId);

        var attachmentId = await ListCommunicationsForCase(caseId);

        await DescribeCaseAttachment(attachmentId);

        await ResolveCase(caseId);

        await DescribeTodayResolvedCases();

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("AWS Support case example scenario complete.");
        Console.WriteLine(new string('-', 80));
    }
    catch (Exception ex)
    {
        logger.LogError(ex, "There was a problem executing the scenario.");
    }
}

/// <summary>
/// List some available services from AWS Support, and select a service for the
example.
/// </summary>
/// <returns>The selected service.</returns>
private static async Task<Service> DisplayAndSelectServices()
{
    Console.WriteLine(new string('-', 80));
    var services = await _supportWrapper.DescribeServices();
    Console.WriteLine($"AWS Support client returned {services.Count} services.");

    Console.WriteLine($"1. Displaying first 10 services:");
    for (int i = 0; i < 10 && i < services.Count; i++)
    {
        Console.WriteLine($" \t{i + 1}. {services[i].Name}");
    }

    var choiceNumber = 0;
    while (choiceNumber < 1 || choiceNumber > services.Count)
    {
        Console.WriteLine(
            "Select an example support service by entering a number from the
preceding list:");
        var choice = Console.ReadLine();
        Int32.TryParse(choice, out choiceNumber);
    }
    Console.WriteLine(new string('-', 80));

    return services[choiceNumber - 1];
}

/// <summary>
/// List the available categories for a service and select a category for the
example.
/// </summary>
/// <param name="service">Service to use for displaying categories.</param>
/// <returns>The selected category.</returns>
private static Category DisplayAndSelectCategories(Service service)
{
    Console.WriteLine(new string('-', 80));

    Console.WriteLine($"2. Available support categories for Service
\"{service.Name}\":");
}
```

```
        for (int i = 0; i < service.Categories.Count; i++)
    {
        Console.WriteLine($"\\t{i + 1}. {service.Categories[i].Name}");
    }

    var choiceNumber = 0;
    while (choiceNumber < 1 || choiceNumber > service.Categories.Count)
    {
        Console.WriteLine(
            "Select an example support category by entering a number from the
preceding list:");
        var choice = Console.ReadLine();
        Int32.TryParse(choice, out choiceNumber);
    }

    Console.WriteLine(new string('-', 80));

    return service.Categories[choiceNumber - 1];
}

/// <summary>
/// List available severity levels from AWS Support, and select a level for the
example.
/// </summary>
/// <returns>The selected severity level.</returns>
private static async Task<SeverityLevel> DisplayAndSelectSeverity()
{
    Console.WriteLine(new string('-', 80));
    var severityLevels = await _supportWrapper.DescribeSeverityLevels();

    Console.WriteLine($"3. Get and display available severity levels:");
    for (int i = 0; i < 10 && i < severityLevels.Count; i++)
    {
        Console.WriteLine($"\\t{i + 1}. {severityLevels[i].Name}");
    }

    var choiceNumber = 0;
    while (choiceNumber < 1 || choiceNumber > severityLevels.Count)
    {
        Console.WriteLine(
            "Select an example severity level by entering a number from the
preceding list:");
        var choice = Console.ReadLine();
        Int32.TryParse(choice, out choiceNumber);
    }

    Console.WriteLine(new string('-', 80));

    return severityLevels[choiceNumber - 1];
}

/// <summary>
/// Create an example support case.
/// </summary>
/// <param name="service">Service to use for the new case.</param>
/// <param name="category">Category to use for the new case.</param>
/// <param name="severity">Severity to use for the new case.</param>
/// <returns>The caseId of the new support case.</returns>
private static async Task<string> CreateSupportCase(Service service,
    Category category, SeverityLevel severity)
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"4. Create an example support case" +
        $" with the following settings:" +
        $" \\n\\tService: {service.Name}, Category: {category.Name}
" +
        $"and Severity Level: {severity.Name}.");
}
```

```
        var caseId = await _supportWrapper.CreateCase(service.Code, category.Code,
severity.Code,
                    "Example case for testing, ignore.", "This is my example support
case.");
                    Console.WriteLine($"\\tNew case created with ID {caseId}");
                    Console.WriteLine(new string('-', 80));
                    return caseId;
    }

    ///<summary>
    /// List open cases for the current day.
    ///</summary>
    ///<returns>Async task.</returns>
private static async Task DescribeTodayOpenCases()
{
    Console.WriteLine($"5. List the open support cases for the current day.");
    // Describe the cases. If it is empty, try again and allow time for the new
    case to appear.
    List<CaseDetails> currentOpenCases = null!;
    while (currentOpenCases == null || currentOpenCases.Count == 0)
    {
        Thread.Sleep(1000);
        currentOpenCases = await _supportWrapper.DescribeCases(
            new List<string>(),
            null,
            false,
            false,
            DateTime.Today,
            DateTime.Now);
    }

    foreach (var openCase in currentOpenCases)
    {
        Console.WriteLine($"\\tCase: {openCase.CaseId} created
{openCase.TimeCreated}");
    }

    Console.WriteLine(new string('-', 80));
}

    ///<summary>
    /// Create an attachment set for a support case.
    ///</summary>
    ///<returns>The attachment set id.</returns>
private static async Task<string> CreateAttachmentSet()
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"6. Create an attachment set for a support case.");
    var fileName = "example_attachment.txt";

    // Create the file if it does not already exist.
    if (!File.Exists(fileName))
    {
        await using StreamWriter sw = File.CreateText(fileName);
        await sw.WriteLineAsync(
                    "This is a sample file for attachment to a support case.");
    }

    await using var ms = new MemoryStream(await
File.ReadAllBytesAsync(fileName));

    var attachmentSetId = await _supportWrapper.AddAttachmentToSet(
        ms,
```

```

        fileName);

        Console.WriteLine($"\\tNew attachment set created with id: \\n
\\t{attachmentSetId.Substring(0, 65)}...");

        Console.WriteLine(new string('-', 80));

        return attachmentSetId;
    }

    ///<summary>
    /// Add an attachment set and communication to a case.
    /// </summary>
    /// <param name="attachmentSetId">Id of the attachment set.</param>
    /// <param name="caseId">Id of the case to receive the attachment set.</param>
    /// <returns>Async task.</returns>
    private static async Task AddCommunicationToCase(string attachmentSetId, string
caseId)
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"7. Add attachment set and communication to {caseId}.");

    await _supportWrapper.AddCommunicationToCase(
        caseId,
        "This is an example communication added to a support case.",
        attachmentSetId);

    Console.WriteLine($"\\tNew attachment set and communication added to
{caseId}");

    Console.WriteLine(new string('-', 80));
}

    ///<summary>
    /// List the communications for a case.
    /// </summary>
    /// <param name="caseId">Id of the case to describe.</param>
    /// <returns>An attachment id.</returns>
    private static async Task<string> ListCommunicationsForCase(string caseId)
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"8. List communications for case {caseId}.");

    var communications = await _supportWrapper.DescribeCommunications(caseId);
    var attachmentId = "";
    foreach (var communication in communications)
    {
        Console.WriteLine(
            $"\\tCommunication created on: {communication.TimeCreated} has
{communication.AttachmentSet.Count} attachments.");
        if (communication.AttachmentSet.Any())
        {
            attachmentId = communication.AttachmentSet.First().AttachmentId;
        }
    }

    Console.WriteLine(new string('-', 80));
    return attachmentId;
}

    ///<summary>
    /// Describe an attachment by id.
    /// </summary>
    /// <param name="attachmentId">Id of the attachment to describe.</param>
    /// <returns>Async task.</returns>
    private static async Task DescribeCaseAttachment(string attachmentId)

```

```
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"9. Describe the attachment set.");

    var attachment = await _supportWrapper.DescribeAttachment(attachmentId);
    var data = Encoding.ASCII.GetString(attachment.Data.ToArray());
    Console.WriteLine($"\\tAttachment includes {attachment.FileName} with data:
\n\\t{data}");

    Console.WriteLine(new string('-', 80));
}

/// <summary>
/// Resolve the support case.
/// </summary>
/// <param name="caseId">Id of the case to resolve.</param>
/// <returns>Async task.</returns>
private static async Task ResolveCase(string caseId)
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"10. Resolve case {caseId}.");

    var status = await _supportWrapper.ResolveCase(caseId);
    Console.WriteLine($"\\tCase {caseId} has final status {status}");

    Console.WriteLine(new string('-', 80));
}

/// <summary>
/// List resolved cases for the current day.
/// </summary>
/// <returns>Async Task.</returns>
private static async Task DescribeTodayResolvedCases()
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"11. List the resolved support cases for the current
day.");
    var currentCases = await _supportWrapper.DescribeCases(
        new List<string>(),
        null,
        false,
        true,
        DateTime.Today,
        DateTime.Now);

    foreach (var currentCase in currentCases)
    {
        if (currentCase.Status == "resolved")
        {
            Console.WriteLine(
                $"\\tCase: {currentCase.CaseId}: status {currentCase.Status}");
        }
    }

    Console.WriteLine(new string('-', 80));
}
}
```

Wrapper methods used by the scenario for AWS Support actions.

```
/// <summary>
/// Wrapper methods to use AWS Support for working with support cases.
/// </summary>
```

```
public class SupportWrapper
{
    private readonly IAmazonAWSSupport _amazonSupport;
    public SupportWrapper(IAmazonAWSSupport amazonSupport)
    {
        _amazonSupport = amazonSupport;
    }

    /// <summary>
    /// Get the descriptions of AWS services.
    /// </summary>
    /// <param name="name">Optional language for services.
    /// Currently "en" (English) and "ja" (Japanese) are supported.</param>
    /// <returns>The list of AWS service descriptions.</returns>
    public async Task<List<Service>> DescribeServices(string language = "en")
    {
        var response = await _amazonSupport.DescribeServicesAsync(
            new DescribeServicesRequest()
            {
                Language = language
            });
        return response.Services;
    }

    /// <summary>
    /// Get the descriptions of support severity levels.
    /// </summary>
    /// <param name="name">Optional language for severity levels.
    /// Currently "en" (English) and "ja" (Japanese) are supported.</param>
    /// <returns>The list of support severity levels.</returns>
    public async Task<List<SeverityLevel>> DescribeSeverityLevels(string language =
"en")
    {
        var response = await _amazonSupport.DescribeSeverityLevelsAsync(
            new DescribeSeverityLevelsRequest()
            {
                Language = language
            });
        return response.SeverityLevels;
    }

    /// <summary>
    /// Create a new support case.
    /// </summary>
    /// <param name="serviceCode">Service code for the new case.</param>
    /// <param name="categoryCode">Category for the new case.</param>
    /// <param name="severityCode">Severity code for the new case.</param>
    /// <param name="subject">Subject of the new case.</param>
    /// <param name="body">Body text of the new case.</param>
    /// <param name="language">Optional language support for your case.
    /// Currently "en" (English) and "ja" (Japanese) are supported.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set for the new
    case.</param>
    /// <param name="issueType">Optional issue type for the new case. Options are
    "customer-service" or "technical".</param>
    /// <returns>The caseId of the new support case.</returns>
    public async Task<string> CreateCase(string serviceCode, string categoryCode,
    string severityCode, string subject,
        string body, string language = "en", string? attachmentSetId = null, string
    issueType = "customer-service")
    {
```

```
        var response = await _amazonSupport.CreateCaseAsync(
            new CreateCaseRequest()
            {
                ServiceCode = serviceCode,
                CategoryCode = categoryCode,
                SeverityCode = severityCode,
                Subject = subject,
                Language = language,
                AttachmentSetId = attachmentSetId,
                IssueType = issueType,
                CommunicationBody = body
            });
        return response.CaseId;
    }

    /// <summary>
    /// Add an attachment to a set, or create a new attachment set if one does not
    /// exist.
    /// </summary>
    /// <param name="data">The data for the attachment.</param>
    /// <param name="fileName">The file name for the attachment.</param>
    /// <param name="attachmentSetId">Optional setId for the attachment. Creates a
    /// new attachment set if empty.</param>
    /// <returns>The setId of the attachment.</returns>
    public async Task<string> AddAttachmentToSet(MemoryStream data, string
fileName, string? attachmentSetId = null)
    {
        var response = await _amazonSupport.AddAttachmentsToSetAsync(
            new AddAttachmentsToSetRequest()
            {
                AttachmentSetId = attachmentSetId,
                Attachments = new List<Attachment>
                {
                    new Attachment
                    {
                        Data = data,
                        FileName = fileName
                    }
                }
            });
        return response.AttachmentSetId;
    }

    /// <summary>
    /// Get description of a specific attachment.
    /// </summary>
    /// <param name="attachmentId">Id of the attachment, usually fetched by
    /// describing the communications of a case.</param>
    /// <returns>The attachment object.</returns>
    public async Task<Attachment> DescribeAttachment(string attachmentId)
    {
        var response = await _amazonSupport.DescribeAttachmentAsync(
            new DescribeAttachmentRequest()
            {
                AttachmentId = attachmentId
            });
        return response.Attachment;
    }

    /// <summary>
```

```

    /// Add communication to a case, including optional attachment set ID and CC
    email addresses.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
    /// <param name="body">Body text of the communication.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set.</param>
    /// <param name="ccEmailAddresses">Optional list of CC email addresses.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> AddCommunicationToCase(string caseId, string body,
        string? attachmentSetId = null, List<string>? ccEmailAddresses = null)
    {
        var response = await _amazonSupport.AddCommunicationToCaseAsync(
            new AddCommunicationToCaseRequest()
            {
                CaseId = caseId,
                CommunicationBody = body,
                AttachmentSetId = attachmentSetId,
                CcEmailAddresses = ccEmailAddresses
            });
        return response.Result;
    }

    /// <summary>
    /// Describe the communications for a case, optionally with a date filter.
    /// </summary>
    /// <param name="caseId">The ID of the support case.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</param>
    /// <param name="beforeTime">The optional end date for a filtered search.</param>
    /// <returns>The list of communications for the case.</returns>
    public async Task<List<Communication>> DescribeCommunications(string caseId,
        DateTime? afterTime = null, DateTime? beforeTime = null)
    {
        var results = new List<Communication>();
        var paginateCommunications =
            _amazonSupport.Paginator.DescribeCommunications(
                new DescribeCommunicationsRequest()
                {
                    CaseId = caseId,
                    AfterTime = afterTime?.ToString("G"),
                    BeforeTime = beforeTime?.ToString("G")
                });
        // Get the entire list using the paginator.
        await foreach (var communications in paginateCommunications.Communications)
        {
            results.Add(communications);
        }
        return results;
    }

    /// <summary>
    /// Get case details for a list of case ids, optionally with date filters.
    /// </summary>
    /// <param name="caseIds">The list of case IDs.</param>
    /// <param name="displayId">Optional display ID.</param>
    /// <param name="includeCommunication">True to include communication. Defaults
    to true.</param>
    /// <param name="includeResolvedCases">True to include resolved cases. Defaults
    to false.</param>
    /// <param name="afterTime">The optional start date for a filtered search.</param>

```

```

    /// <param name="beforeTime">The optional end date for a filtered search.</param>
    /// <param name="language">Optional language support for your case.
    /// Currently "en" (English) and "ja" (Japanese) are supported.</param>
    /// <returns>A list of CaseDetails.</returns>
    public async Task<List<CaseDetails>> DescribeCases(List<string> caseIds,
    string? displayId = null, bool includeCommunication = true,
    bool includeResolvedCases = false, DateTime? afterTime = null, DateTime?
    beforeTime = null,
    string language = "en")
    {
        var results = new List<CaseDetails>();
        var paginateCases = _amazonSupport.Paginator.DescribeCases(
            new DescribeCasesRequest()
        {
            CaseIdList = caseIds,
            DisplayId = displayId,
            IncludeCommunications = includeCommunication,
            IncludeResolvedCases = includeResolvedCases,
            AfterTime = afterTime?.ToString("o"),
            BeforeTime = beforeTime?.ToString("o"),
            Language = language
        });
        // Get the entire list using the paginator.
        await foreach (var cases in paginateCases.Cases)
        {
            results.Add(cases);
        }
        return results;
    }

    /// <summary>
    /// Resolve a support case by caseId.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
    /// <returns>The final status of the case after resolving.</returns>
    public async Task<string> ResolveCase(string caseId)
    {
        var response = await _amazonSupport.ResolveCaseAsync(
            new ResolveCaseRequest()
        {
            CaseId = caseId
        });
        return response.FinalCaseStatus;
    }

    /// <summary>
    /// Verify the support level for AWS Support API access.
    /// </summary>
    /// <returns>True if the subscription level supports API access.</returns>
    public async Task<bool> VerifySubscription()
    {
        try
        {
            var response = await _amazonSupport.DescribeServicesAsync(
                new DescribeServicesRequest()
            {
                Language = "en"
            });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (Amazon.AWSSupport.AmazonAWSSupportException ex)
        {
    
```

```
        if (ex.ErrorCode == "SubscriptionRequiredException")
    {
        return false;
    }
    else throw;
}
}
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [AddAttachmentsToSet](#)
  - [AddCommunicationToCase](#)
  - [CreateCase](#)
  - [DescribeAttachment](#)
  - [DescribeCases](#)
  - [DescribeCommunications](#)
  - [DescribeServices](#)
  - [DescribeSeverityLevels](#)
  - [ResolveCase](#)

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run various AWS Support operations.

```
/**
 * Before running this Java (v2) code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS Support
 * Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets and displays available services.
 * 2. Gets and displays severity levels.
 * 3. Creates a support case by using the selected service, category, and severity
 * level.
 * 4. Gets a list of open cases for the current day.
 * 5. Creates an attachment set with a generated file.
 * 6. Adds a communication with the attachment to the support case.
 * 7. Lists the communications of the support case.
 * 8. Describes the attachment set included with the communication.
 * 9. Resolves the support case.
 * 10. Gets a list of resolved cases for the current day.
 */
public class SupportScenario {
```

```
    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "      <fileAttachment>" +
            "Where:\n" +
            "      fileAttachment - The file can be a simple saved .txt file to use
as an email attachment. \n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String fileAttachment = args[0];
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("***** Welcome to the AWS Support case example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Get and display available services.");
        List<String> sevCatList = displayServices(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get and display Support severity levels.");
        String sevLevel = displaySevLevels(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a support case using the selected service,
category, and severity level.");
        String caseId = createSupportCase(supportClient, sevCatList, sevLevel);
        if (caseId.compareTo("")==0) {
            System.out.println("A support case was not successfully created!");
            System.exit(1);
        } else
            System.out.println("Support case "+caseId +" was successfully
created!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get open support cases.");
        getOpenCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create an attachment set with a generated file to
add to the case.");
        String attachmentSetId = addAttachment(supportClient, fileAttachment);
        System.out.println("The Attachment Set id value is" +attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add communication with the attachment to the support
case.");
        addAttachSupportCase(supportClient, caseId, attachmentSetId);
        System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("7. List the communications of the support case.");
String attachId = listCommunications(supportClient, caseId);
System.out.println("The Attachment id value is" +attachId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Describe the attachment set included with the
communication.");
describeAttachment(supportClient, attachId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Resolve the support case.");
resolveSupportCase(supportClient, caseId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a list of resolved cases for the current
day.");
getResolvedCase(supportClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("***** This Scenario has successfully completed");
System.out.println(DASHES);
}

public static void getResolvedCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
    .maxResults(30)
    .afterTime(yesterday.toString())
    .beforeTime(now.toString())
    .includeResolvedCases(true)
    .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase: cases) {
            if (sinCase.status().compareTo("resolved") ==0)
                System.out.println("The case status is "+sinCase.status());
        }
    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
        .caseId(caseId)
        .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
```

```
        System.out.println("The status of case "+caseId +" is
"+response.finalCaseStatus());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeAttachment(SupportClient supportClient, String
attachId) {
    try {
        DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
            .attachmentId(attachId)
            .build();

        DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
        System.out.println("The name of the file is
"+response.attachment().fileName());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static String listCommunications(SupportClient supportClient, String
caseId) {
    try {
        String attachId = null;
        DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
            .caseId(caseId)
            .maxResults(10)
            .build();

        DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
        List<Communication> communications = response.communications();
        for (Communication comm: communications) {
            System.out.println("the body is: " + comm.body());

            //Get the attachment id value.
            List<AttachmentDetails> attachments = comm.attachmentSet();
            for (AttachmentDetails detail : attachments) {
                attachId = detail.attachmentId();
            }
        }
        return attachId;
    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
            .caseId(caseId)
            .attachmentSetId(attachmentSetId)
```

```
.communicationBody("Please refer to attachment for details.")
.build();

AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
if (response.result())
    System.out.println("You have successfully added a communication to
an AWS Support case");
else
    System.out.println("There was an error adding the communication to
an AWS Support case");

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
try {
    File myFile = new File(fileAttachment);
    InputStream sourceStream = new FileInputStream(myFile);
    SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

    Attachment attachment = Attachment.builder()
        .fileName(myFile.getName())
        .data(sourceBytes)
        .build();

    AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
        .attachments(attachment)
        .build();

    AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
    return response.attachmentSetId();

} catch (SupportException | FileNotFoundException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return "";
}

public static void getOpenCase(SupportClient supportClient) {
try {
    // Specify the start and end time.
    Instant now = Instant.now();
    java.time.LocalDate.now();
    Instant yesterday = now.minus(1, ChronoUnit.DAYS);

    DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
        .maxResults(20)
        .afterTime(yesterday.toString())
        .beforeTime(now.toString())
        .build();

    DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
    List<CaseDetails> cases = response.cases();
    for (CaseDetails sinCase: cases) {
        System.out.println("The case status is "+sinCase.status());
        System.out.println("The case Id is "+sinCase.caseId());
    }
}
}
```

```
        System.out.println("The case subject is "+sinCase.subject());
    }

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

public static String createSupportCase(SupportClient supportClient,
List<String> sevCatList, String sevLevel) {
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
            .severityCode(sevLevel.toLowerCase())
            .communicationBody("Test issue with "+serviceCode.toLowerCase())
            .subject("Test case, please ignore")
            .language("en")
            .issueType("technical")
            .build();

        CreateCaseResponse response = supportClient.createCase(caseRequest);
        return response.caseId();
    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static String displaySevLevels(SupportClient supportClient) {
    try {
        DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
    .language("en")
    .build();

        DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
        List<SeverityLevel> severityLevels = response.severityLevels();
        String levelName = null;
        for (SeverityLevel sevLevel: severityLevels) {
            System.out.println("The severity level name is: "+
sevLevel.name());
            if (sevLevel.name().compareTo("High")==0)
                levelName = sevLevel.name();
        }
        return levelName;
    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
    .language("en")
```

```
.build();

DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
String serviceCode = null;
String catName = null;
List<String> sevCatList = new ArrayList<>();
List<Service> services = response.services();

System.out.println("Get the first 10 services");
int index = 1;
for (Service service: services) {
    if (index== 11)
        break;

    System.out.println("The Service name is: "+service.name());
    if (service.name().compareTo("Account") == 0)
        serviceCode = service.code();

    // Get the Categories for this service.
    List<Category> categories = service.categories();
    for (Category cat: categories) {
        System.out.println("The category name is: "+cat.name());
        if (cat.name().compareTo("Security") == 0)
            catName = cat.name();
    }
    index++ ;
}

// Push the two values to the list.
sevCatList.add(serviceCode);
sevCatList.add(catName);
return sevCatList;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return null;
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [AddAttachmentsToSet](#)
  - [AddCommunicationToCase](#)
  - [CreateCase](#)
  - [DescribeAttachment](#)
  - [DescribeCases](#)
  - [DescribeCommunications](#)
  - [DescribeServices](#)
  - [DescribeSeverityLevels](#)
  - [ResolveCase](#)

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
Before running this Kotlin code example, set up your development environment,  
including your credentials.  
  
For more information, see the following documentation topic:  
  
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html  
In addition, you must have the AWS Business Support Plan to use the AWS Support  
Java API. For more information, see:  
  
https://aws.amazon.com/premiumsupport/plans/  
  
This Kotlin example performs the following tasks:  
1. Gets and displays available services.  
2. Gets and displays severity levels.  
3. Creates a support case by using the selected service, category, and severity  
level.  
4. Gets a list of open cases for the current day.  
5. Creates an attachment set with a generated file.  
6. Adds a communication with the attachment to the support case.  
7. Lists the communications of the support case.  
8. Describes the attachment set included with the communication.  
9. Resolves the support case.  
10. Gets a list of resolved cases for the current day.  
*/  
  
suspend fun main(args: Array<String>) {  
    val usage = """  
Usage:  
      <fileAttachment>  
Where:  
      fileAttachment - The file can be a simple saved .txt file to use as an  
email attachment.  
    """  
  
    if (args.size != 1) {  
        println(usage)  
        exitProcess(0)  
    }  
  
    val fileAttachment = args[0]  
    println("***** Welcome to the AWS Support case example scenario.")  
    println("***** Step 1. Get and display available services.")  
    val sevCatList = displayServices()  
  
    println("***** Step 2. Get and display Support severity levels.")  
    val sevLevel = displaySevLevels()  
  
    println("***** Step 3. Create a support case using the selected service,  
category, and severity level.")  
    val caseIdVal = createSupportCase(sevCatList, sevLevel)  
    if (caseIdVal != null) {
```

```
        println("Support case $caseIdVal was successfully created!")
    } else {
        println("A support case was not successfully created!")
        exitProcess(1)
    }

    println("***** Step 4. Get open support cases.")
getOpenCase()

println("***** Step 5. Create an attachment set with a generated file to add to
the case.")
val attachmentSetId = addAttachment(fileAttachment)
println("The Attachment Set id value is $attachmentSetId")

println("***** Step 6. Add communication with the attachment to the support
case.")
addAttachSupportCase(caseIdVal, attachmentSetId)

println("***** Step 7. List the communications of the support case.")
val attachId = listCommunications(caseIdVal)
println("The Attachment id value is $attachId")

println("***** Step 8. Describe the attachment set included with the
communication.")
describeAttachment(attachId)

println("***** Step 9. Resolve the support case.")
resolveSupportCase(caseIdVal)

println("***** Step 10. Get a list of resolved cases for the current day.")
getResolvedCase()
println("***** This Scenario has successfully completed")
}

suspend fun getResolvedCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest = DescribeCasesRequest {
        maxResults = 30
        afterTime = yesterday.toString()
        beforeTime = now.toString()
        includeResolvedCases = true
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest = ResolveCaseRequest {
        caseId = caseIdVal
    }
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}
```

```
suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest = DescribeAttachmentRequest {
        attachmentId = attachId
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}

suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest = DescribeCommunicationsRequest {
        caseId = caseIdVal
        maxResults = 10
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}

suspend fun addAttachSupportCase(caseIdVal: String?, attachmentSetIdVal: String?) {
    val caseRequest = AddCommunicationToCaseRequest {
        caseId = caseIdVal
        attachmentSetId = attachmentSetIdVal
        communicationBody = "Please refer to attachment for details."
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS Support
case")
        } else {
            println("There was an error adding the communication to an AWS Support
case")
        }
    }
}

suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal = Attachment {
        fileName = myFile.name
        data = sourceBytes
    }

    val setRequest = AddAttachmentsToSetRequest {
        attachments = listOf(attachmentVal)
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}
```

```
suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest = DescribeCasesRequest {
        maxResults = 20
        afterTime = yesterday.toString()
        beforeTime = now.toString()
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun createSupportCase(sevCatListVal: List<String>, sevLevelVal: String): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest = CreateCaseRequest {
        categoryCode = caseCategory.lowercase(Locale.getDefault())
        serviceCode = serCode.lowercase(Locale.getDefault())
        severityCode = sevLevelVal.lowercase(Locale.getDefault())
        communicationBody = "Test issue with
${serCode.lowercase(Locale.getDefault())}"
        subject = "Test case, please ignore"
        language = "en"
        issueType = "technical"
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}

suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest = DescribeSeverityLevelsRequest {
        language = "en"
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
        return levelName
    }
}

// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest = DescribeServicesRequest {
```

```
        language = "en"
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }
        }

        // Get the categories for this service.
        service.categories?.forEach { cat ->
            println("The category name is ${cat.name}")
            if (cat.name == "Security") {
                catName = cat.name!!
            }
        }
        index++
    }
}

// Push the two values to the list.
serviceCode.let { sevCatList.add(it) }
catName.let { sevCatList.add(it) }
return sevCatList
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - [AddAttachmentsToSet](#)
  - [AddCommunicationToCase](#)
  - [CreateCase](#)
  - [DescribeAttachment](#)
  - [DescribeCases](#)
  - [DescribeCommunications](#)
  - [DescribeServices](#)
  - [DescribeSeverityLevels](#)
  - [ResolveCase](#)

## Code examples for Systems Manager using AWS SDKs

The following code examples show how to use AWS Systems Manager with an AWS software development kit (SDK).

### Code examples

- [Actions for Systems Manager using AWS SDKs \(p. 2201\)](#)
  - [Add a Systems Manager parameter using an AWS SDK \(p. 2201\)](#)

- Get information about Systems Manager parameters using an AWS SDK (p. 2201)

## Actions for Systems Manager using AWS SDKs

The following code examples show how to use AWS Systems Manager with AWS SDKs. Each example calls an individual service function.

### Examples

- Add a Systems Manager parameter using an AWS SDK (p. 2201)
- Get information about Systems Manager parameters using an AWS SDK (p. 2201)

## Add a Systems Manager parameter using an AWS SDK

The following code example shows how to add a Systems Manager parameter.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_parameter(
    client: &Client,
    name: &str,
    value: &str,
    description: &str,
) -> Result<(), Error> {
    let resp = client
        .put_parameter()
        .overwrite(true)
        .r#type(ParameterType::String)
        .name(name)
        .value(value)
        .description(description)
        .send()
        .await?;

    println!("Success! Parameter now has version: {}", resp.version());
    Ok(())
}
```

- For API details, see [PutParameter](#) in *AWS SDK for Rust API reference*.

## Get information about Systems Manager parameters using an AWS SDK

The following code example shows how to get Systems Manager parameters information.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_parameters(client: &Client) -> Result<(), Error> {
    let resp = client.describe_parameters().send().await?;

    for param in resp.parameters().unwrap().iter() {
        println!("  {}", param.name().unwrap_or_default());
    }

    Ok(())
}
```

- For API details, see [DescribeParameters](#) in *AWS SDK for Rust API reference*.

## Code examples for Amazon Textract using AWS SDKs

The following code examples show how to use Amazon Textract with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon Textract using AWS SDKs \(p. 2202\)](#)
  - Analyze a document using Amazon Textract and an AWS SDK (p. 2203)
  - Detect text in a document using Amazon Textract and an AWS SDK (p. 2206)
  - Get data about an Amazon Textract document analysis job using an AWS SDK (p. 2209)
  - Start asynchronous analysis of a document using Amazon Textract and an AWS SDK (p. 2210)
  - Start asynchronous text detection using Amazon Textract and an AWS SDK (p. 2214)
- [Scenarios for Amazon Textract using AWS SDKs \(p. 2216\)](#)
  - Get started with Amazon Textract document analysis using an AWS SDK (p. 2216)
- [Cross-service examples for Amazon Textract using AWS SDKs \(p. 2218\)](#)
  - Create an Amazon Textract explorer application (p. 2218)
  - Detect entities in text extracted from an image using an AWS SDK (p. 2219)

## Actions for Amazon Textract using AWS SDKs

The following code examples show how to use Amazon Textract with AWS SDKs. Each example calls an individual service function.

### Examples

- [Analyze a document using Amazon Textract and an AWS SDK \(p. 2203\)](#)
- [Detect text in a document using Amazon Textract and an AWS SDK \(p. 2206\)](#)
- [Get data about an Amazon Textract document analysis job using an AWS SDK \(p. 2209\)](#)
- [Start asynchronous analysis of a document using Amazon Textract and an AWS SDK \(p. 2210\)](#)
- [Start asynchronous text detection using Amazon Textract and an AWS SDK \(p. 2214\)](#)

## Analyze a document using Amazon Textract and an AWS SDK

The following code examples show how to analyze a document using Amazon Textract.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void analyzeDoc(TextractClient textractClient, String sourceDoc) {
    try {
        InputStream sourceStream = new FileInputStream(new File(sourceDoc));
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Get the input Document object as bytes
        Document myDoc = Document.builder()
            .bytes(sourceBytes)
            .build();

        List<FeatureType> featureTypes = new ArrayList<FeatureType>();
        featureTypes.add(FeatureType.FORMS);
        featureTypes.add(FeatureType.TABLES);

        AnalyzeDocumentRequest analyzeDocumentRequest =
        AnalyzeDocumentRequest.builder()
            .featureTypes(featureTypes)
            .document(myDoc)
            .build();

        AnalyzeDocumentResponse analyzeDocument =
        textractClient.analyzeDocument(analyzeDocumentRequest);
        List<Block> docInfo = analyzeDocument.blocks();
        Iterator<Block> blockIterator = docInfo.iterator();

        while(blockIterator.hasNext()) {
            Block block = blockIterator.next();
            System.out.println("The block type is "
            +block.blockType().toString());
        }

    } catch (TextractException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [AnalyzeDocument](#) in [AWS SDK for Java 2.x API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class TextractWrapper:  
    """Encapsulates Textract functions."""  
    def __init__(self, textract_client, s3_resource, sqs_resource):  
        """  
        :param textract_client: A Boto3 Textract client.  
        :param s3_resource: A Boto3 Amazon S3 resource.  
        :param sqs_resource: A Boto3 Amazon SQS resource.  
        """  
        self.textract_client = textract_client  
        self.s3_resource = s3_resource  
        self.sqs_resource = sqs_resource  
  
    def analyze_file(  
        self, feature_types, *, document_file_name=None, document_bytes=None):  
        """  
        Detects text and additional elements, such as forms or tables, in a local  
        image file or from in-memory byte data.  
        The image must be in PNG or JPG format.  
  
        :param feature_types: The types of additional document features to detect.  
        :param document_file_name: The name of a document image file.  
        :param document_bytes: In-memory byte data of a document image.  
        :return: The response from Amazon Textract, including a list of blocks  
                that describe elements detected in the image.  
        """  
        if document_file_name is not None:  
            with open(document_file_name, 'rb') as document_file:  
                document_bytes = document_file.read()  
        try:  
            response = self.textract_client.analyze_document(  
                Document={'Bytes': document_bytes}, FeatureTypes=feature_types)  
            logger.info(  
                "Detected %s blocks.", len(response['Blocks']))  
        except ClientError:  
            logger.exception("Couldn't detect text.")  
            raise  
        else:  
            return response
```

- For API details, see [AnalyzeDocument](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

"Detects text and additional elements, such as forms or tables,
"in a local image file or from in-memory byte data."
"The image must be in PNG or JPG format."

DATA lo_document TYPE REF TO /aws1/cl_txtdocument.
DATA lo_s3object TYPE REF TO /aws1/cl_txts3object.
DATA lo_featuretypes TYPE REF TO /aws1/cl_txfeaturetypes_w.
DATA lt_featuretypes TYPE /aws1/cl_txfeaturetypes_w=>tt_featuretypes.

"Create ABAP objects for feature type"
"add TABLES to return information about the tables"
"add FORMS to return detected form data."
"To perform both types of analysis, add TABLES and FORMS to FeatureTypes"

CREATE OBJECT lo_featuretypes EXPORTING iv_value = 'FORMS'.
INSERT lo_featuretypes INTO TABLE lt_featuretypes.

CREATE OBJECT lo_featuretypes EXPORTING iv_value = 'TABLES'.
INSERT lo_featuretypes INTO TABLE lt_featuretypes.

"Create an ABAP object for the S3 obejct."
CREATE OBJECT lo_s3object
  EXPORTING
    iv_bucket = iv_s3bucket
    iv_name   = iv_s3object.

"Create an ABAP object for the document."
CREATE OBJECT lo_document EXPORTING io_s3object = lo_s3object.

"Analyze document stored in S3"
TRY.
  oo_result = lo_tx->analyzedocument(          "oo_result is returned for
testing purpose"
  EXPORTING
    io_document      = lo_document
    it_featuretypes = lt_featuretypes
  ).
  MESSAGE 'Analyze document completed' TYPE 'I'.
CATCH /aws1/cx_txaccessdeniedex .
  MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
CATCH /aws1/cx_txbaddocumentex .
  MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
CATCH /aws1/cx_txdocumenttoolargeex .
  MESSAGE 'The document is too large' TYPE 'E'.
CATCH /aws1/cx_txhlquotaexceededex .
  MESSAGE 'Human loop quota has been exceeded' TYPE 'E'.
CATCH /aws1/cx_txinternalservererr .
  MESSAGE 'Internal server error' TYPE 'E'.
CATCH /aws1/cx_txinvalidparameterex .
  MESSAGE 'Request has invalid parameters' TYPE 'E'.
CATCH /aws1/cx_txinvalids3objectex .
  MESSAGE 'S3 object is invalid' TYPE 'E'.
CATCH /aws1/cx_txprovthruputexcdex .
  MESSAGE 'Provisioned throughput exceeded limit' TYPE 'E'.
CATCH /aws1/cx_txthrottlingex .
  MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
CATCH /aws1/cx_txunsupporteddocex .
  MESSAGE 'The document is not supported' TYPE 'E'.
ENDTRY.

```

- For API details, see [AnalyzeDocument](#) in *AWS SDK for SAP ABAP API reference*.

## Detect text in a document using Amazon Textract and an AWS SDK

The following code examples show how to detect text in a document using Amazon Textract.

Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Detect text from an input document.

```
public static void detectDocText(TextractClient textractClient, String sourceDoc) {  
  
    try {  
        InputStream sourceStream = new FileInputStream(new File(sourceDoc));  
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);  
  
        // Get the input Document object as bytes  
        Document myDoc = Document.builder()  
            .bytes(sourceBytes)  
            .build();  
  
        DetectDocumentTextRequest detectDocumentTextRequest =  
        DetectDocumentTextRequest.builder()  
            .document(myDoc)  
            .build();  
  
        // Invoke the Detect operation  
        DetectDocumentTextResponse textResponse =  
        textractClient.detectDocumentText(detectDocumentTextRequest);  
        List<Block> docInfo = textResponse.blocks();  
        for (Block block : docInfo) {  
            System.out.println("The block type is " +  
                block.blockType().toString());  
        }  
  
        DocumentMetadata documentMetadata = textResponse.documentMetadata();  
        System.out.println("The number of pages in the document is "  
            +documentMetadata.pages());  
  
    } catch (TextractException | FileNotFoundException e) {  
  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

Detect text from a document located in an Amazon S3 bucket.

```
public static void detectDocTextS3 (TextractClient textractClient, String bucketName, String docName) {  
  
    try {  
        S3Object s3Object = S3Object.builder()  
            .bucket(bucketName)  
            .name(docName)
```

```
.build();

// Create a Document object and reference the s3Object instance
Document myDoc = Document.builder()
    .s3Object(s3Object)
    .build();

DetectDocumentTextRequest detectDocumentTextRequest =
DetectDocumentTextRequest.builder()
    .document(myDoc)
    .build();

DetectDocumentTextResponse textResponse =
textextractClient.detectDocumentText(detectDocumentTextRequest);
for (Block block: textResponse.blocks()) {
    System.out.println("The block type is "
+block.blockType().toString());
}

DocumentMetadata documentMetadata = textResponse.documentMetadata();
System.out.println("The number of pages in the document is "
+documentMetadata.pages());

} catch (TextextractException e) {

    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [DetectDocumentText in AWS SDK for Java 2.x API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class TextextractWrapper:
    """Encapsulates Textextract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textextract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def detect_file_text(self, *, document_file_name=None, document_bytes=None):
        """
        Detects text elements in a local image file or from in-memory byte data.
        The image must be in PNG or JPG format.

        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textextract, including a list of blocks
                that describe elements detected in the image.
        """

```

```
if document_file_name is not None:
    with open(document_file_name, 'rb') as document_file:
        document_bytes = document_file.read()
try:
    response = self.textract_client.detect_document_text(
        Document={'Bytes': document_bytes})
    logger.info(
        "Detected %s blocks.", len(response['Blocks']))
except ClientError:
    logger.exception("Couldn't detect text.")
    raise
else:
    return response
```

- For API details, see [DetectDocumentText in AWS SDK for Python \(Boto3\) API Reference](#).

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Detects text in the input document."
"Amazon Textract can detect lines of text and the words that make up a line of
text."
"The input document must be in one of the following image formats: JPEG, PNG,
PDF, or TIFF."

DATA lo_document TYPE REF TO /aws1/cl_txdocument.
DATA lo_s3object TYPE REF TO /aws1/cl_txs3object.

"Create an ABAP object for the S3 obejct."
CREATE OBJECT lo_s3object
  EXPORTING
    iv_bucket = iv_s3bucket
    iv_name   = iv_s3object.

"Create an ABAP object for the document."
CREATE OBJECT lo_document EXPORTING io_s3object = lo_s3object.

"Analyze document stored in S3"
TRY.
  oo_result = lo_tx->detectdocumenttext( io_document = lo_document ).
"oo_result is returned for testing purpose"
  MESSAGE 'Detect document text completed' TYPE 'I'.
  CATCH /aws1/cx_txaccessdeniedex .
    MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
  CATCH /aws1/cx_txbaddocumentex .
    MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
  CATCH /aws1/cx_txdocumenttoolargeex .
    MESSAGE 'The document is too large' TYPE 'E'.
  CATCH /aws1/cx_txinternalservererr .
    MESSAGE 'Internal server error' TYPE 'E'.
  CATCH /aws1/cx_txinvalidparameterex .
    MESSAGE 'Request has invalid parameters' TYPE 'E'.
```

```
CATCH /aws1/cx_txinvalids3objectex .
    MESSAGE 'S3 object is invalid' TYPE 'E'.
CATCH /aws1/cx_txprovthruputexcdex .
    MESSAGE 'Provisioned throughput exceeded limit' TYPE 'E'.
CATCH /aws1/cx_txthrottlingex .
    MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
CATCH /aws1/cx_txunSupportedDocEx .
    MESSAGE 'The document is not supported' TYPE 'E'.
ENDTRY.
```

- For API details, see [DetectDocumentText in AWS SDK for SAP ABAP API reference](#).

## Get data about an Amazon Textract document analysis job using an AWS SDK

The following code examples show how to get data about an Amazon Textract document analysis job.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def get_analysis_job(self, job_id):
        """
        Gets data for a previously started detection job that includes additional
        elements.

        :param job_id: The ID of the job to retrieve.
        :return: The job data, including a list of blocks that describe elements
                detected in the image.
        """
        try:
            response = self.textract_client.get_document_analysis(
                JobId=job_id)
            job_status = response['JobStatus']
            logger.info("Job %s status is %s.", job_id, job_status)
        except ClientError:
            logger.exception("Couldn't get data for job %s.", job_id)
            raise
        else:
            return response
```

- For API details, see [GetDocumentAnalysis in AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

## SDK for SAP ABAP

### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Gets the results for an Amazon Textract"
"asynchronous operation that analyzes text in a document."
TRY.
    oo_result = lo_tex->getdocumentanalysis( iv_jobid = iv_jobid ).
"oo_result is returned for testing purpose"
    MESSAGE 'Document analysis retrieved' TYPE 'I'.
    CATCH /aws1/cx_texaccessdenieddex .
        MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
    CATCH /aws1/cx_texinternalservererr .
        MESSAGE 'Internal server error' TYPE 'E'.
    CATCH /aws1/cx_txetinvalidjobidex .
        MESSAGE 'Job ID is invalid' TYPE 'E'.
    CATCH /aws1/cx_txetinvalidkmskeyex .
        MESSAGE 'KMS key is invalid' TYPE 'E'.
    CATCH /aws1/cx_txetinvalidparameterex .
        MESSAGE 'Request has invalid parameters' TYPE 'E'.
    CATCH /aws1/cx_txetinvalids3objectex .
        MESSAGE 'S3 object is invalid' TYPE 'E'.
    CATCH /aws1/cx_txetprovthruputexcdex .
        MESSAGE 'Provisioned throughput exceeded limit' TYPE 'E'.
    CATCH /aws1/cx_txetrottlingex .
        MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
ENDTRY.
```

- For API details, see [GetDocumentAnalysis in AWS SDK for SAP ABAP API reference](#).

## Start asynchronous analysis of a document using Amazon Textract and an AWS SDK

The following code examples show how to start asynchronous analysis of a document using Amazon Textract.

Java

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String startDocAnalysisS3 (TextractClient textractClient, String
bucketName, String docName) {

    try {
```

```
List<FeatureType> myList = new ArrayList<>();
myList.add(FeatureType.TABLES);
myList.add(FeatureType.FORMS);

S3Object s3Object = S3Object.builder()
    .bucket(bucketName)
    .name(docName)
    .build();

DocumentLocation location = DocumentLocation.builder()
    .s3Object(s3Object)
    .build();

StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
    .documentLocation(location)
    .featureTypes(myList)
    .build();

StartDocumentAnalysisResponse response =
textractClient.startDocumentAnalysis(documentAnalysisRequest);

// Get the job ID
String jobId = response.jobId();
return jobId;

} catch (TextractException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "" ;
}

private static String getJobResults(TextractClient textractClient, String
jobId) {

boolean finished = false;
int index = 0 ;
String status = "" ;

try {
    while (!finished) {
        GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
        .jobId(jobId)
        .maxResults(1000)
        .build();

        GetDocumentAnalysisResponse response =
textractClient.getDocumentAnalysis(analysisRequest);
        status = response.jobStatus().toString();

        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(index + " status is: " + status);
            Thread.sleep(1000);
        }
        index++ ;
    }
    return status;
} catch( InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
```

```
        }
    return "";
}
```

- For API details, see [StartDocumentAnalysis](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Start an asynchronous job to analyze a document.

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def start_analysis_job(
            self, bucket_name, document_file_name, feature_types, sns_topic_arn,
            sns_role_arn):
        """
        Starts an asynchronous job to detect text and additional elements, such as
        forms or tables, in an image stored in an Amazon S3 bucket. Textract
        publishes
        a notification to the specified Amazon SNS topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the
        image.
        :param document_file_name: The name of the document image stored in Amazon
        S3.
        :param feature_types: The types of additional document features to detect.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS topic
        where job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management (IAM)
        role that can be assumed by Textract and grants
        permission
            to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_analysis(
                DocumentLocation={
                    'S3Object': {'Bucket': bucket_name, 'Name': document_file_name},
                    'NotificationChannel={
                        'SNSTopicArn': sns_topic_arn, 'RoleArn': sns_role_arn},
                    'FeatureTypes=feature_types}
                }
            job_id = response['JobId']
            logger.info(
                "Started text analysis job %s on %s.", job_id, document_file_name)
        except Exception as e:
            logger.error("Error starting text analysis job: %s", str(e))
            raise e
    
```

```
        except ClientError:
            logger.exception("Couldn't analyze text in %s.", document_file_name)
            raise
        else:
            return job_id
```

- For API details, see [StartDocumentAnalysis](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Starts the asynchronous analysis of an input document for relationships"
"between detected items such as key-value pairs, tables, and selection
elements."

DATA lo_documentlocation TYPE REF TO /aws1/cl_txdocumentlocation.
DATA lo_s3object TYPE REF TO /aws1/cl_txs3object.
DATA lo_featuretypes TYPE REF TO /aws1/cl_txfeaturetypes_w.
DATA lt_featuretypes TYPE /aws1/cl_txfeaturetypes_w=>tt_featuretypes.

"Create ABAP objects for feature type"
"add TABLES to return information about the tables"
"add FORMS to return detected form data."
"To perform both types of analysis, add TABLES and FORMS to FeatureTypes"

CREATE OBJECT lo_featuretypes EXPORTING iv_value = 'FORMS'.
INSERT lo_featuretypes INTO TABLE lt_featuretypes.

CREATE OBJECT lo_featuretypes EXPORTING iv_value = 'TABLES'.
INSERT lo_featuretypes INTO TABLE lt_featuretypes.

"Create an ABAP object for the S3 obejct."
CREATE OBJECT lo_s3object
    EXPORTING
        iv_bucket = iv_s3bucket
        iv_name   = iv_s3object.

"Create an ABAP object for the document."
CREATE OBJECT lo_documentlocation EXPORTING io_s3object = lo_s3object.

"Start async document analysis."
TRY.
    oo_result = lo_tx->startdocumentanalysis(          "oo_result is returned for
testing purpose"
        EXPORTING
            io_documentlocation      = lo_documentlocation
            it_featuretypes         = lt_featuretypes
        ).
    MESSAGE 'Document analysis started' TYPE 'I'.
CATCH /aws1/cx_txaccessdeniedex .
    MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
CATCH /aws1/cx_txbaddocumentex .
```

```
MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
CATCH /aws1/cx_txdocumenttoolargeex .
    MESSAGE 'The document is too large' TYPE 'E'.
CATCH /aws1/cx_txidempotentprmmis00 .
    MESSAGE 'Idempotent parameter mismatch exception' TYPE 'E'.
CATCH /aws1/cx_txinternalservererr .
    MESSAGE 'Internal server error' TYPE 'E'.
CATCH /aws1/cx_txinvalidkmskeyex .
    MESSAGE 'KMS key is invalid' TYPE 'E'.
CATCH /aws1/cx_txinvalidparameterex .
    MESSAGE 'Request has invalid parameters' TYPE 'E'.
CATCH /aws1/cx_txinvalids3objectex .
    MESSAGE 'S3 object is invalid' TYPE 'E'.
CATCH /aws1/cx_txlimitexceededex .
    MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
CATCH /aws1/cx_txprovthruputexcdex .
    MESSAGE 'Provisioned throughput exceeded limit' TYPE 'E'.
CATCH /aws1/cx_txthrottlingex .
    MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
CATCH /aws1/cx_txunsupporteddocex .
    MESSAGE 'The document is not supported' TYPE 'E'.
ENDTRY.
```

- For API details, see [StartDocumentAnalysis](#) in *AWS SDK for SAP ABAP API reference*.

## Start asynchronous text detection using Amazon Textract and an AWS SDK

The following code examples show how to start asynchronous text detection in a document using Amazon Textract.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Start an asynchronous job to detect text in a document.

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def start_detection_job(
            self, bucket_name, document_file_name, sns_topic_arn, sns_role_arn):
        """
        Starts an asynchronous job to detect text elements in an image stored in an
        Amazon S3 bucket. Textract publishes a notification to the specified Amazon
        topic when the job completes.
        
```

```
The image must be in PNG, JPG, or PDF format.

:param bucket_name: The name of the Amazon S3 bucket that contains the
image.
:param document_file_name: The name of the document image stored in Amazon
S3.
:param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS topic
where the job completion notification is published.
:param sns_role_arn: The ARN of an AWS Identity and Access Management (IAM)
role that can be assumed by Textract and grants
permission
    to publish to the Amazon SNS topic.
:return: The ID of the job.
"""
try:
    response = self.textract_client.start_document_text_detection(
        DocumentLocation={
            'S3Object': {'Bucket': bucket_name, 'Name': document_file_name}},
        NotificationChannel={
            'SNSTopicArn': sns_topic_arn, 'RoleArn': sns_role_arn})
    job_id = response['JobId']
    logger.info(
        "Started text detection job %s on %s.", job_id, document_file_name)
except ClientError:
    logger.exception("Couldn't detect text in %s.", document_file_name)
    raise
else:
    return job_id
```

- For API details, see [StartDocumentTextDetection](#) in *AWS SDK for Python (Boto3) API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Starts the asynchronous detection of text in a document."
"Amazon Textract can detect lines of text and the words that make up a line of
text."
DATA lo_documentlocation TYPE REF TO /aws1/cl_txtdocumentlocation.
DATA lo_s3object TYPE REF TO /aws1/cl_txts3object.

"Create an ABAP object for the S3 obejct."
CREATE OBJECT lo_s3object
    EXPORTING
        iv_bucket = iv_s3bucket
        iv_name   = iv_s3object.

"Create an ABAP object for the document."
CREATE OBJECT lo_documentlocation
    EXPORTING
        io_s3object = lo_s3object.
```

```
"Start document analysis."
TRY.
    oo_result = lo_tx->startdocumenttextdetection( io_documentlocation =
lo_documentlocation ).           "oo_result is returned for testing purpose"
    MESSAGE 'Document analysis started' TYPE 'I'.
    CATCH /aws1/cx_txaccessdeniedex .
        MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
    CATCH /aws1/cx_txbaddocumentex .
        MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
    CATCH /aws1/cx_txdocumenttoolargeex .
        MESSAGE 'The document is too large' TYPE 'E'.
    CATCH /aws1/cx_txidempotentprmmis00 .
        MESSAGE 'Idempotent parameter mismatch exception' TYPE 'E'.
    CATCH /aws1/cx_txinternalservererr .
        MESSAGE 'Internal server error' TYPE 'E'.
    CATCH /aws1/cx_txinvalidkmskeyex .
        MESSAGE 'KMS key is invalid' TYPE 'E'.
    CATCH /aws1/cx_txinvalidparameterex .
        MESSAGE 'Request has invalid parameters' TYPE 'E'.
    CATCH /aws1/cx_txinvalids3objectex .
        MESSAGE 'S3 object is invalid' TYPE 'E'.
    CATCH /aws1/cx_txlimitexceededex .
        MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
    CATCH /aws1/cx_txprovthruputexcdex .
        MESSAGE 'Provisioned throughput exceeded limit' TYPE 'E'.
    CATCH /aws1/cx_txthrottlingex .
        MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
    CATCH /aws1/cx_txunsupporteddocex .
        MESSAGE 'The document is not supported' TYPE 'E'.
ENDTRY.
```

- For API details, see [StartDocumentTextDetection](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios for Amazon Textract using AWS SDKs

The following code examples show how to use Amazon Textract with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Get started with Amazon Textract document analysis using an AWS SDK \(p. 2216\)](#)

## Get started with Amazon Textract document analysis using an AWS SDK

The following code example shows how to:

- Start asynchronous analysis.
- Get document analysis.

SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_documentlocation TYPE REF TO /aws1/cl_txdocumentlocation.  
DATA lo_s3object TYPE REF TO /aws1/cl_txs3object.  
DATA lo_featuretypes TYPE REF TO /aws1/cl_txfeaturetypes_w.  
DATA lt_featuretypes TYPE /aws1/cl_txfeaturetypes_w=>tt_featuretypes.  
  
"Create ABAP objects for feature type"  
"add TABLES to return information about the tables"  
"add FORMS to return detected form data."  
"To perform both types of analysis, add TABLES and FORMS to FeatureTypes"  
  
CREATE OBJECT lo_featuretypes EXPORTING iv_value = 'FORMS'.  
INSERT lo_featuretypes INTO TABLE lt_featuretypes.  
  
CREATE OBJECT lo_featuretypes EXPORTING iv_value = 'TABLES'.  
INSERT lo_featuretypes INTO TABLE lt_featuretypes.  
  
"Create an ABAP object for the S3 obejct."  
CREATE OBJECT lo_s3object  
    EXPORTING  
        iv_bucket = iv_s3bucket  
        iv_name   = iv_s3object.  
  
"Create an ABAP object for the document."  
CREATE OBJECT lo_documentlocation EXPORTING io_s3object = lo_s3object.  
  
"Start document analysis."  
TRY.  
    DATA(lo_start_result) = lo_tx->startdocumentanalysis(  
        EXPORTING  
            io_documentlocation      = lo_documentlocation  
            it_featuretypes         = lt_featuretypes  
        ).  
    MESSAGE 'Document analysis started' TYPE 'I'.  
    CATCH /aws1/cx_txaccessdeniedex .  
        MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.  
    CATCH /aws1/cx_txbaddocumentex .  
        MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.  
    CATCH /aws1/cx_txdocumenttoolargeex .  
        MESSAGE 'The document is too large' TYPE 'E'.  
    CATCH /aws1/cx_txidempotentprmmis00 .  
        MESSAGE 'Idempotent parameter mismatch exception' TYPE 'E'.  
    CATCH /aws1/cx_txinternalservererr .  
        MESSAGE 'Internal server error' TYPE 'E'.  
    CATCH /aws1/cx_txinvalidkmskeyex .  
        MESSAGE 'KMS key is invalid' TYPE 'E'.  
    CATCH /aws1/cx_txinvalidparameterex .  
        MESSAGE 'Request has invalid parameters' TYPE 'E'.  
    CATCH /aws1/cx_txinvalids3objectex .  
        MESSAGE 'S3 object is invalid' TYPE 'E'.  
    CATCH /aws1/cx_txlimitexceededex .  
        MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.  
    CATCH /aws1/cx_txprovthruputexcdex .  
        MESSAGE 'Provisioned throughput exceeded limit' TYPE 'E'.  
    CATCH /aws1/cx_txthrottlingex .  
        MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.  
    CATCH /aws1/cx_txunsupporteddocex .  
        MESSAGE 'The document is not supported' TYPE 'E'.  
ENDTRY.
```

```
"Get job ID from the output"
DATA(lv_jobid) = lo_start_result->get_jobid( ).

"Wait for job to complete"
oo_result = lo_tex->getdocumentanalysis( EXPORTING iv_jobid = lv_jobid ).      "
oo_result is returned for testing purpose "
WHILE oo_result->get_jobstatus( ) <> 'SUCCEEDED'.
  IF sy-index = 10.
    EXIT.                      "maximum 300 seconds"
  ENDIF.
  WAIT UP TO 30 SECONDS.
  oo_result = lo_tex->getdocumentanalysis( EXPORTING iv_jobid = lv_jobid )."
ENDWHILE.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [GetDocumentAnalysis](#)
  - [StartDocumentAnalysis](#)

## Cross-service examples for Amazon Textract using AWS SDKs

The following code examples show how to use Amazon Textract with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Create an Amazon Textract explorer application \(p. 2218\)](#)
- [Detect entities in text extracted from an image using an AWS SDK \(p. 2219\)](#)

## Create an Amazon Textract explorer application

The following code examples show how to explore Amazon Textract output through an interactive application.

JavaScript

### SDK for JavaScript V3

Shows how to use the AWS SDK for JavaScript to build a React application that uses Amazon Textract to extract data from a document image and display it in an interactive web page. This example runs in a web browser and requires an authenticated Amazon Cognito identity for credentials. It uses Amazon Simple Storage Service (Amazon S3) for storage, and for notifications it polls an Amazon Simple Queue Service (Amazon SQS) queue that is subscribed to an Amazon Simple Notification Service (Amazon SNS) topic.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS

- Amazon Textract

Python

#### **SDK for Python (Boto3)**

Shows how to use the AWS SDK for Python (Boto3) with Amazon Textract to detect text, form, and table elements in a document image. The input image and Amazon Textract output are shown in a Tkinter application that lets you explore the detected elements.

- Submit a document image to Amazon Textract and explore the output of detected elements.
- Submit images directly to Amazon Textract or through an Amazon Simple Storage Service (Amazon S3) bucket.
- Use asynchronous APIs to start a job that publishes a notification to an Amazon Simple Notification Service (Amazon SNS) topic when the job completes.
- Poll an Amazon Simple Queue Service (Amazon SQS) queue for a job completion message and display the results.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

## Detect entities in text extracted from an image using an AWS SDK

The following code example shows how to use Amazon Comprehend to detect entities in text extracted by Amazon Textract from an image that is stored in Amazon S3.

Python

#### **SDK for Python (Boto3)**

Shows how to use the AWS SDK for Python (Boto3) in a Jupyter notebook to detect entities in text that is extracted from an image. This example uses Amazon Textract to extract text from an image stored in Amazon Simple Storage Service (Amazon S3) and Amazon Comprehend to detect entities in the extracted text.

This example is a Jupyter notebook and must be run in an environment that can host notebooks. For instructions on how to run the example using Amazon SageMaker, see the directions in [TextractAndComprehendNotebook.ipynb](#).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### **Services used in this example**

- Amazon Comprehend
- Amazon S3
- Amazon Textract

# Code examples for Amazon Transcribe using AWS SDKs

The following code examples show how to use Amazon Transcribe with an AWS software development kit (SDK).

## Code examples

- [Actions for Amazon Transcribe using AWS SDKs \(p. 2220\)](#)
  - [Create a custom Amazon Transcribe vocabulary using an AWS SDK \(p. 2221\)](#)
  - [Delete a custom Amazon Transcribe vocabulary using an AWS SDK \(p. 2222\)](#)
  - [Delete a Amazon Transcribe Medical transcription job using an AWS SDK \(p. 2223\)](#)
  - [Delete an Amazon Transcribe transcription job using an AWS SDK \(p. 2224\)](#)
  - [Get a custom Amazon Transcribe vocabulary using an AWS SDK \(p. 2226\)](#)
  - [Get an Amazon Transcribe transcription job using an AWS SDK \(p. 2227\)](#)
  - [List custom Amazon Transcribe vocabularies using an AWS SDK \(p. 2228\)](#)
  - [List Amazon Transcribe Medical transcription jobs using an AWS SDK \(p. 2229\)](#)
  - [List Amazon Transcribe transcription jobs using an AWS SDK \(p. 2231\)](#)
  - [Produce real-time transcriptions with Amazon Transcribe using an AWS SDK \(p. 2233\)](#)
  - [Start an Amazon Transcribe Medical transcription job using an AWS SDK \(p. 2236\)](#)
  - [Start an Amazon Transcribe transcription job using an AWS SDK \(p. 2238\)](#)
  - [Update a custom Amazon Transcribe vocabulary using an AWS SDK \(p. 2240\)](#)
- [Scenarios for Amazon Transcribe using AWS SDKs \(p. 2241\)](#)
  - [Create and refine an Amazon Transcribe custom vocabulary using an AWS SDK \(p. 2242\)](#)
  - [Transcribe audio and get job data with Amazon Transcribe using an AWS SDK \(p. 2247\)](#)
- [Cross-service examples for Amazon Transcribe using AWS SDKs \(p. 2248\)](#)
  - [Build an Amazon Transcribe app \(p. 2248\)](#)
  - [Build an Amazon Transcribe streaming app \(p. 2249\)](#)
  - [Convert text to speech and back to text using an AWS SDK \(p. 2249\)](#)

## Actions for Amazon Transcribe using AWS SDKs

The following code examples show how to use Amazon Transcribe with AWS SDKs. Each example calls an individual service function.

## Examples

- [Create a custom Amazon Transcribe vocabulary using an AWS SDK \(p. 2221\)](#)
- [Delete a custom Amazon Transcribe vocabulary using an AWS SDK \(p. 2222\)](#)
- [Delete a Amazon Transcribe Medical transcription job using an AWS SDK \(p. 2223\)](#)
- [Delete an Amazon Transcribe transcription job using an AWS SDK \(p. 2224\)](#)
- [Get a custom Amazon Transcribe vocabulary using an AWS SDK \(p. 2226\)](#)
- [Get an Amazon Transcribe transcription job using an AWS SDK \(p. 2227\)](#)
- [List custom Amazon Transcribe vocabularies using an AWS SDK \(p. 2228\)](#)
- [List Amazon Transcribe Medical transcription jobs using an AWS SDK \(p. 2229\)](#)
- [List Amazon Transcribe transcription jobs using an AWS SDK \(p. 2231\)](#)
- [Produce real-time transcriptions with Amazon Transcribe using an AWS SDK \(p. 2233\)](#)
- [Start an Amazon Transcribe Medical transcription job using an AWS SDK \(p. 2236\)](#)

- Start an Amazon Transcribe transcription job using an AWS SDK (p. 2238)
- Update a custom Amazon Transcribe vocabulary using an AWS SDK (p. 2240)

## Create a custom Amazon Transcribe vocabulary using an AWS SDK

The following code examples show how to create a custom Amazon Transcribe vocabulary.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create a custom vocabulary using a list of phrases. Custom vocabularies
/// improve transcription accuracy for one or more specific words.
/// </summary>
/// <param name="languageCode">The language code of the vocabulary.</param>
/// <param name="phrases">Phrases to use in the vocabulary.</param>
/// <param name="vocabularyName">Name for the vocabulary.</param>
/// <returns>The state of the custom vocabulary.</returns>
public async Task<VocabularyState> CreateCustomVocabulary(LanguageCode
languageCode,
    List<string> phrases, string vocabularyName)
{
    var response = await _amazonTranscribeService.CreateVocabularyAsync(
        new CreateVocabularyRequest
    {
        LanguageCode = languageCode,
        Phrases = phrases,
        VocabularyName = vocabularyName
    });
    return response.VocabularyState;
}
```

- For API details, see [CreateVocabulary](#) in *AWS SDK for .NET API Reference*.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_vocabulary(
    vocabulary_name, language_code, transcribe_client,
    phrases=None, table_uri=None):
    """
Creates a custom vocabulary that can be used to improve the accuracy of
transcription jobs. This function returns as soon as the vocabulary processing
is started. Call get_vocabulary to get the current status of the vocabulary.
```

```
The vocabulary is ready to use when its status is 'READY'.
```

```
:param vocabulary_name: The name of the custom vocabulary.  
:param language_code: The language code of the vocabulary.  
    For example, en-US or nl-NL.  
:param transcribe_client: The Boto3 Transcribe client.  
:param phrases: A list of comma-separated phrases to include in the vocabulary.  
:param table_uri: A table of phrases and pronunciation hints to include in the vocabulary.  
:return: Information about the newly created vocabulary.  
"""  
try:  
    vocab_args = {'VocabularyName': vocabulary_name, 'LanguageCode':  
language_code}  
    if phrases is not None:  
        vocab_args['Phrases'] = phrases  
    elif table_uri is not None:  
        vocab_args['VocabularyFileUri'] = table_uri  
    response = transcribe_client.create_vocabulary(**vocab_args)  
    logger.info("Created custom vocabulary %s.", response['VocabularyName'])  
except ClientError:  
    logger.exception("Couldn't create custom vocabulary %s.", vocabulary_name)  
    raise  
else:  
    return response
```

- For API details, see [CreateVocabulary](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a custom Amazon Transcribe vocabulary using an AWS SDK

The following code examples show how to delete a custom Amazon Transcribe vocabulary.

### .NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Delete an existing custom vocabulary.  
/// </summary>  
/// <param name="vocabularyName">Name of the vocabulary to delete.</param>  
/// <returns>True if successful.</returns>  
public async Task<bool> DeleteCustomVocabulary(string vocabularyName)  
{  
    var response = await _amazonTranscribeService.DeleteVocabularyAsync(  
        new DeleteVocabularyRequest  
    {  
        VocabularyName = vocabularyName  
    });  
    return response.StatusCode == HttpStatusCode.OK;  
}
```

- For API details, see [DeleteVocabulary](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

- For API details, see [DeleteVocabulary](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a Amazon Transcribe Medical transcription job using an AWS SDK

The following code examples show how to delete an Amazon Transcribe Medical transcription job.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Delete a medical transcription job. Also deletes the transcript associated
with the job.
/// </summary>
/// <param name="jobName">Name of the medical transcription job to delete.</
param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteMedicalTranscriptionJob(string jobName)
{
    var response = await
_amazonTranscribeService.DeleteMedicalTranscriptionJobAsync(
    new DeleteMedicalTranscriptionJobRequest()
    {
        MedicalTranscriptionJobName = jobName
    });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- For API details, see [DeleteMedicalTranscriptionJob](#) in *AWS SDK for .NET API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Delete a medical transcription job.

```
// Import the required AWS SDK clients and commands for Node.js
import { DeleteMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // For example,
  'medical_transcription_demo'
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new DeleteMedicalTranscriptionJobCommand(params)
    );
    console.log("Success - deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteMedicalTranscriptionJob](#) in [AWS SDK for JavaScript API Reference](#).

## Delete an Amazon Transcribe transcription job using an AWS SDK

The following code examples show how to delete an Amazon Transcribe transcription job.

### .NET

#### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    /// <summary>
    /// Delete a transcription job. Also deletes the transcript associated with the
    job.
    /// </summary>
    /// <param name="jobName">Name of the transcription job to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTranscriptionJob(string jobName)
    {
        var response = await _amazonTranscribeService.DeleteTranscriptionJobAsync(
            new DeleteTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- For API details, see [DeleteTranscriptionJob](#) in *AWS SDK for .NET API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Delete a transcription job.

```
// Import the required AWS SDK clients and commands for Node.js
import { DeleteTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
    TranscriptionJobName: "JOB_NAME", // Required. For example, 'transcription_demo'
};

export const run = async () => {
    try {
        const data = await transcribeClient.send(
            new DeleteTranscriptionJobCommand(params)
        );
        console.log("Success - deleted");
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteTranscriptionJob](#) in [AWS SDK for JavaScript API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_job(job_name, transcribe_client):
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.

    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_transcription_job(
            TranscriptionJobName=job_name)
        logger.info("Deleted job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't delete job %s.", job_name)
        raise
```

- For API details, see [DeleteTranscriptionJob](#) in [AWS SDK for Python \(Boto3\) API Reference](#).

## Get a custom Amazon Transcribe vocabulary using an AWS SDK

The following code examples show how to get a custom Amazon Transcribe vocabulary.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get information about a custom vocabulary.
/// </summary>
/// <param name="vocabularyName">Name of the vocabulary.</param>
/// <returns>The state of the custom vocabulary.</returns>
public async Task<VocabularyState> GetCustomVocabulary(string vocabularyName)
{
    var response = await _amazonTranscribeService.GetVocabularyAsync(
        new GetVocabularyRequest()
    {
        VocabularyName = vocabularyName
    });
    return response.VocabularyState;
```

```
}
```

- For API details, see [GetVocabulary](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_vocabulary(vocabulary_name, transcribe_client):
    """
    Gets information about a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: Information about the vocabulary.
    """
    try:
        response = transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Got vocabulary %s.", response['VocabularyName'])
    except ClientError:
        logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
        raise
    else:
        return response
```

- For API details, see [GetVocabulary](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get an Amazon Transcribe transcription job using an AWS SDK

The following code examples show how to get an Amazon Transcribe transcription job.

### .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get details about a transcription job.
/// </summary>
/// <param name="jobName">A unique name for the transcription job.</param>
/// <returns>A TranscriptionJob instance with information on the requested
job.</returns>
public async Task<TranscriptionJob> GetTranscriptionJob(string jobName)
{
    var response = await _amazonTranscribeService.GetTranscriptionJobAsync(
        new GetTranscriptionJobRequest()
    {
        TranscriptionJobName = jobName
    });
    return response.TranscriptionJob;
}
```

```
        });
    return response.TranscriptionJob;
}
```

- For API details, see [GetTranscriptionJob](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The retrieved transcription job.
    """
    try:
        response = transcribe_client.get_transcription_job(
            TranscriptionJobName=job_name)
        job = response['TranscriptionJob']
        logger.info("Got job %s.", job['TranscriptionJobName'])
    except ClientError:
        logger.exception("Couldn't get job %s.", job_name)
        raise
    else:
        return job
```

- For API details, see [GetTranscriptionJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## List custom Amazon Transcribe vocabularies using an AWS SDK

The following code examples show how to list custom Amazon Transcribe vocabularies.

### .NET

#### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// List custom vocabularies for the current account. Optionally specify a name
/// filter and a specific state to filter the vocabularies list.
/// </summary>
/// <param name="nameContains">Optional string the vocabulary name must
/// contain.</param>
/// <param name="stateEquals">Optional state of the vocabulary.</param>
/// <returns>List of information about the vocabularies.</returns>
```

```
public async Task<List<VocabularyInfo>> ListCustomVocabularies(string?  
nameContains = null,  
        VocabularyState? stateEquals = null)  
{  
    var response = await _amazonTranscribeService.ListVocabulariesAsync(  
        new ListVocabulariesRequest()  
    {  
        NameContains = nameContains,  
        StateEquals = stateEquals  
    });  
    return response.Vocabularies;  
}
```

- For API details, see [ListVocabularies](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_vocabularies(vocabulary_filter, transcribe_client):  
    """  
    Lists the custom vocabularies created for this AWS account.  
  
    :param vocabulary_filter: The returned vocabularies must contain this string in  
                            their names.  
    :param transcribe_client: The Boto3 Transcribe client.  
    :return: The list of retrieved vocabularies.  
    """  
    try:  
        response = transcribe_client.list_vocabularies(  
            NameContains=vocabulary_filter)  
        vocabs = response['Vocabularies']  
        next_token = response.get('NextToken')  
        while next_token is not None:  
            response = transcribe_client.list_vocabularies(  
                NameContains=vocabulary_filter, NextToken=next_token)  
            vocabs += response['Vocabularies']  
            next_token = response.get('NextToken')  
        logger.info(  
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter)  
    except ClientError:  
        logger.exception(  
            "Couldn't list vocabularies with filter %s.", vocabulary_filter)  
        raise  
    else:  
        return vocabs
```

- For API details, see [ListVocabularies](#) in *AWS SDK for Python (Boto3) API Reference*.

## List Amazon Transcribe Medical transcription jobs using an AWS SDK

The following code examples show how to list Amazon Transcribe Medical transcription jobs.

## .NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// List medical transcription jobs, optionally with a name filter.
/// </summary>
/// <param name="jobNameContains">Optional name filter for the medical
transcription jobs.</param>
/// <returns>A list of summaries about medical transcription jobs.</returns>
public async Task<List<MedicalTranscriptionJobSummary>>
ListMedicalTranscriptionJobs(
    string? jobNameContains = null)
{
    var response = await
_amazonTranscribeService.ListMedicalTranscriptionJobsAsync(
    new ListMedicalTranscriptionJobsRequest()
    {
        JobNameContains = jobNameContains
    });
    return response.MedicalTranscriptionJobSummaries;
}
```

- For API details, see [ListMedicalTranscriptionJobs](#) in *AWS SDK for .NET API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

List medical transcription jobs.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
    MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
    OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
```

```
Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'  
Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and 'DICTATION'  
LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'  
MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'  
Media: {  
    MediaFileUri: "SOURCE_FILE_LOCATION",  
    // The S3 object location of the input media file. The URI must be in the same  
    region  
    // as the API endpoint that you are calling. For example,  
    // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"  
},  
};  
  
export const run = async () => {  
    try {  
        const data = await transcribeClient.send(  
            new StartMedicalTranscriptionJobCommand(params)  
        );  
        console.log("Success - put", data);  
        return data; // For unit tests.  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListMedicalTranscriptionJobs](#) in [AWS SDK for JavaScript API Reference](#).

## List Amazon Transcribe transcription jobs using an AWS SDK

The following code examples show how to list Amazon Transcribe transcription jobs.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// List transcription jobs, optionally with a name filter.  
/// </summary>  
/// <param name="jobNameContains">Optional name filter for the transcription  
jobs.</param>  
/// <returns>A list of transcription job summaries.</returns>  
public async Task<List<TranscriptionJobSummary>> ListTranscriptionJobs(string?  
jobNameContains = null)  
{  
    var response = await _amazonTranscribeService.ListTranscriptionJobsAsync(  
        new ListTranscriptionJobsRequest()  
    {  
        JobNameContains = jobNameContains  
    });  
    return response.TranscriptionJobSummaries;  
}
```

- For API details, see [ListTranscriptionJobs](#) in *AWS SDK for .NET API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

List transcription jobs.

```
// Import the required AWS SDK clients and commands for Node.js
import { ListTranscriptionJobsCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  JobNameContains: "KEYWORD", // Not required. Returns only transcription
  // job names containing this string
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new ListTranscriptionJobsCommand(params)
    );
    console.log("Success", data.TranscriptionJobSummaries);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListTranscriptionJobs](#) in *AWS SDK for JavaScript API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_jobs(job_filter, transcribe_client):
```

```
"""
Lists summaries of the transcription jobs for the current AWS account.

:param job_filter: The list of returned jobs must contain this string in their
                    names.
:param transcribe_client: The Boto3 Transcribe client.
:return: The list of retrieved transcription job summaries.
"""
try:
    response = transcribe_client.list_transcription_jobs(
        JobNameContains=job_filter)
    jobs = response['TranscriptionJobSummaries']
    next_token = response.get('NextToken')
    while next_token is not None:
        response = transcribe_client.list_transcription_jobs(
            JobNameContains=job_filter, NextToken=next_token)
        jobs += response['TranscriptionJobSummaries']
        next_token = response.get('NextToken')
    logger.info("Got %s jobs with filter %s.", len(jobs), job_filter)
except ClientError:
    logger.exception("Couldn't get jobs with filter %s.", job_filter)
    raise
else:
    return jobs
```

- For API details, see [ListTranscriptionJobs](#) in *AWS SDK for Python (Boto3) API Reference*.

## Produce real-time transcriptions with Amazon Transcribe using an AWS SDK

The following code example shows how to produce a real-time transcription with Amazon Transcribe.

C++

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
int main() {
    Aws::SDKOptions options;
    options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;
    Aws::InitAPI(options);
{
    //TODO(User): Set to the region of your AWS account.
    const Aws::String region = Aws::Region::US_WEST_2;

    Aws::Utils::Threading::Semaphore canCloseStream(0 /*initialCount*/, 1 /
*maxCount*/);

    //Load a profile that has been granted AmazonTranscribeFullAccess AWS
    managed permission policy.
    Aws::Client::ClientConfiguration config;
#ifndef _WIN32
    // ATTENTION: On Windows with AWS SDK version 1.9, this example only runs
    if the SDK is built
        // with the curl library. (9/15/2022)
        // For more information, see the accompanying ReadMe.
```

```

// For more information, see "Building the SDK for Windows with curl".
// https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
windows.html
//TODO(User): Update to the location of your .crt file.
config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
config.region = region;

TranscribeStreamingServiceClient client(config);
StartStreamTranscriptionHandler handler;
handler.SetOnErrorCallback(
    [](const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
&error) {
        std::cerr << "ERROR: " + error.GetMessage() << std::endl;
    });
//SetTranscriptEventCallback called for every 'chunk' of file transcribed.
// Partial results are returned in real time.
handler.SetTranscriptEventCallback([&canCloseStream](const TranscriptEvent
&ev) {
    bool isFinal = false;
    for (auto &r: ev.GetTranscript().GetResults()) {
        if (r.GetIsPartial()) {
            std::cout << "[partial] ";
        }
        else {
            std::cout << "[Final] ";
            isFinal = true;
        }
        for (auto &alt: r.GetAlternatives()) {
            std::cout << alt.GetTranscript() << std::endl;
        }
    }
    if (isFinal) {
        canCloseStream.Release();
    }
});

StartStreamTranscriptionRequest request;
request.SetMediaSampleRateHertz(8000);
request.SetLanguageCode(LanguageCode::en_US);
request.SetMediaEncoding(
    MediaEncoding::pcm); // wav and aiff files are PCM formats.
request.SetEventStreamHandler(handler);

auto OnStreamReady = [&canCloseStream](AudioStream &stream) {
    Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
    if (!file.is_open()) {
        std::cerr << "Failed to open " << FILE_NAME << '\n';
    }
    std::array<char, BUFFER_SIZE> buf;
    int i = 0;
    while (file) {
        file.read(&buf[0], buf.size());
        if (!file)
            std::cout << "File: only " << file.gcount() << " could be
read"
            << std::endl;
        Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
        AudioEvent event(std::move(bits));
        if (!stream) {
            std::cerr << "Failed to create a stream" << std::endl;
            break;
        }
    }
}

```

```

//The std::basic_istream::gcount() is used to count the
characters in the given string. It returns
//the number of characters extracted by the last read()
operation.
    if (file.gcount() > 0) {
        if (!istream.WriteAudioEvent(event)) {
            std::cerr << "Failed to write an audio event" <<
std::endl;
            break;
        }
    } else {
        break;
    }
    std::this_thread::sleep_for(std::chrono::milliseconds(
        25)); // Slow down because we are streaming from a
file.
}
if (!istream.WriteAudioEvent(
    AudioEvent())) {
    // Per the spec, we have to send an empty event (an event
without a payload) at the end.
    std::cerr << "Failed to send an empty frame" << std::endl;
}
else {
    std::cout << "Successfully sent the empty frame" << std::endl;
}
stream.flush();
// Wait until the final transcript or an error is received.
// Closing the stream prematurely will trigger an error.
canCloseStream.WaitOne();
stream.Close();
};

Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /
*maxCount*/);
auto OnResponseCallback = [&signaling, &canCloseStream](
    const TranscribeStreamingServiceClient * /*unused*/,
    const Model::StartStreamTranscriptionRequest & /*unused*/,
    const Model::StartStreamTranscriptionOutcome & outcome,
    const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /
*unused*/) {
    if (!outcome.IsSuccess())
    {
        std::cerr << "Transcribe streaming error " <<
outcome.GetError().GetMessage() << std::endl;
    }

    canCloseStream.Release();
    signaling.Release();
};

std::cout << "Starting..." << std::endl;
client.StartStreamTranscriptionAsync(request, OnStreamReady,
OnResponseCallback,
                nullptr /*context*/);
signaling.WaitOne(); // Prevent the application from exiting until we're
done.
std::cout << "Done" << std::endl;
}

Aws::ShutdownAPI(options);

return 0;
}

```

- For API details, see [StartStreamTranscriptionAsync](#) in *AWS SDK for C++ API Reference*.

## Start an Amazon Transcribe Medical transcription job using an AWS SDK

The following code examples show how to start an Amazon Transcribe Medical transcription job.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Start a medical transcription job for a media file. This method returns
/// as soon as the job is started.
/// </summary>
/// <param name="jobName">A unique name for the medical transcription job.</param>
/// <param name="mediaFileUri">The URI of the media file, typically an Amazon
S3 location.</param>
/// <param name="mediaFormat">The format of the media file.</param>
/// <param name="outputBucketName">Location for the output, typically an Amazon
S3 location.</param>
/// <param name="transcriptionType">Conversation or dictation transcription
type.</param>
/// <returns>A MedicalTransactionJob instance with information on the new
job.</returns>
public async Task<MedicalTransactionJob> StartMedicalTranscriptionJob(
    string jobName, string mediaFileUri,
    MediaFormat mediaFormat, string outputBucketName,
Amazon.TranscribeService.Type transcriptionType)
{
    var response = await
_amazonTranscribeService.StartMedicalTranscriptionJobAsync(
    new StartMedicalTranscriptionJobRequest()
    {
        MedicalTranscriptionJobName = jobName,
        Media = new Media()
        {
            MediaFileUri = mediaFileUri
        },
        MediaFormat = mediaFormat,
        LanguageCode =
            LanguageCode
                .EnUS, // The value must be en-US for medical
transcriptions.
        OutputBucketName = outputBucketName,
        OutputKey =
            jobName, // The value is a key used to fetch the output of the
transcription.
        Specialty = Specialty.PRIMARYCARE, // The value PRIMARYCARE must be
set.
        Type = transcriptionType
    });
    return response.MedicalTranscriptionJob;
```

```
}
```

- For API details, see [StartMedicalTranscriptionJob](#) in *AWS SDK for .NET API Reference*.

## JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Start a medical transcription job.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
    MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
    OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
    Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
    Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and 'DICTATION'
    LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
    MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
    Media: {
        MediaFileUri: "SOURCE_FILE_LOCATION",
        // The S3 object location of the input media file. The URI must be in the same
        region
        // as the API endpoint that you are calling. For example,
        // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
    },
};

export const run = async () => {
    try {
        const data = await transcribeClient.send(
            new StartMedicalTranscriptionJobCommand(params)
        );
        console.log("Success - put", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [StartMedicalTranscriptionJob](#) in *AWS SDK for JavaScript API Reference*.

## Start an Amazon Transcribe transcription job using an AWS SDK

The following code examples show how to start an Amazon Transcribe transcription job.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Start a transcription job for a media file. This method returns
/// as soon as the job is started.
/// </summary>
/// <param name="jobName">A unique name for the transcription job.</param>
/// <param name="mediaFileUri">The URI of the media file, typically an Amazon
S3 location.</param>
/// <param name="mediaFormat">The format of the media file.</param>
/// <param name="languageCode">The language code of the media file, such as en-
US.</param>
/// <param name="vocabularyName">Optional name of a custom vocabulary.</param>
/// <returns>A TranscriptionJob instance with information on the new job.</
returns>
public async Task<TranscriptionJob> StartTranscriptionJob(string jobName,
string mediaFileUri,
MediaFormat mediaFormat, LanguageCode languageCode, string? vocabularyName)
{
    var response = await _amazonTranscribeService.StartTranscriptionJobAsync(
        new StartTranscriptionJobRequest()
    {
        TranscriptionJobName = jobName,
        Media = new Media()
        {
            MediaFileUri = mediaFileUri
        },
        MediaFormat = mediaFormat,
        LanguageCode = languageCode,
        Settings = vocabularyName != null ? new Settings()
        {
            VocabularyName = vocabularyName
        } : null
    });
    return response.TranscriptionJob;
}
```

- For API details, see [StartTranscriptionJob](#) in *AWS SDK for .NET API Reference*.

JavaScript

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Start a transcription job.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
    TranscriptionJobName: "JOB_NAME",
    LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
    MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
    Media: {
        MediaFileUri: "SOURCE_LOCATION",
        // For example, "https://transcribe-demo.s3-REGION.amazonaws.com/
hello_world.wav"
    },
    OutputBucketName: "OUTPUT_BUCKET_NAME"
};

export const run = async () => {
    try {
        const data = await transcribeClient.send(
            new StartTranscriptionJobCommand(params)
        );
        console.log("Success - put", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [StartTranscriptionJob in AWS SDK for JavaScript API Reference](#).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def start_job(
    job_name, media_uri, media_format, language_code, transcribe_client,
    vocabulary_name=None):
    """
    Starts a transcription job. This function returns as soon as the job is
    started.
```

To get the current status of the job, call `get_transcription_job`. The job is successfully completed when the job status is 'COMPLETED'.

```
:param job_name: The name of the transcription job. This must be unique for
                  your AWS account.
:param media_uri: The URI where the audio file is stored. This is typically
                  in an Amazon S3 bucket.
:param media_format: The format of the audio file. For example, mp3 or wav.
:param language_code: The language code of the audio file.
                  For example, en-US or ja-JP
:param transcribe_client: The Boto3 Transcribe client.
:param vocabulary_name: The name of a custom vocabulary to use when
transcribing
                  the audio file.
:return: Data about the job.
"""
try:
    job_args = {
        'TranscriptionJobName': job_name,
        'Media': {'MediaFileUri': media_uri},
        'MediaFormat': media_format,
        'LanguageCode': language_code}
    if vocabulary_name is not None:
        job_args['Settings'] = {'VocabularyName': vocabulary_name}
    response = transcribe_client.start_transcription_job(**job_args)
    job = response['TranscriptionJob']
    logger.info("Started transcription job %s.", job_name)
except ClientError:
    logger.exception("Couldn't start transcription job %s.", job_name)
    raise
else:
    return job
```

- For API details, see [StartTranscriptionJob in AWS SDK for Python \(Boto3\) API Reference](#).

## Update a custom Amazon Transcribe vocabulary using an AWS SDK

The following code examples show how to update a custom Amazon Transcribe vocabulary.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Update a custom vocabulary with new values. Update overwrites all existing
information.
/// </summary>
/// <param name="languageCode">The language code of the vocabulary.</param>
/// <param name="phrases">Phrases to use in the vocabulary.</param>
/// <param name="vocabularyName">Name for the vocabulary.</param>
/// <returns>The state of the custom vocabulary.</returns>
public async Task<VocabularyState> UpdateCustomVocabulary(LanguageCode
languageCode,
List<string> phrases, string vocabularyName)
```

```
{  
    var response = await _amazonTranscribeService.UpdateVocabularyAsync(  
        new UpdateVocabularyRequest()  
    {  
        LanguageCode = languageCode,  
        Phrases = phrases,  
        VocabularyName = vocabularyName  
    });  
    return response.VocabularyState;  
}
```

- For API details, see [UpdateVocabulary](#) in *AWS SDK for .NET API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def update_vocabulary(  
    vocabulary_name, language_code, transcribe_client, phrases=None,  
    table_uri=None):  
    """  
    Updates an existing custom vocabulary. The entire vocabulary is replaced with  
    the contents of the update.  
  
    :param vocabulary_name: The name of the vocabulary to update.  
    :param language_code: The language code of the vocabulary.  
    :param transcribe_client: The Boto3 Transcribe client.  
    :param phrases: A list of comma-separated phrases to include in the vocabulary.  
    :param table_uri: A table of phrases and pronunciation hints to include in the  
        vocabulary.  
    """  
    try:  
        vocab_args = {'VocabularyName': vocabulary_name, 'LanguageCode':  
language_code}  
        if phrases is not None:  
            vocab_args['Phrases'] = phrases  
        elif table_uri is not None:  
            vocab_args['VocabularyFileUri'] = table_uri  
        response = transcribe_client.update_vocabulary(**vocab_args)  
        logger.info(  
            "Updated custom vocabulary %s.", response['VocabularyName'])  
    except ClientError:  
        logger.exception("Couldn't update custom vocabulary %s.", vocabulary_name)  
    raise
```

- For API details, see [UpdateVocabulary](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios for Amazon Transcribe using AWS SDKs

The following code examples show how to use Amazon Transcribe with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Create and refine an Amazon Transcribe custom vocabulary using an AWS SDK \(p. 2242\)](#)
- [Transcribe audio and get job data with Amazon Transcribe using an AWS SDK \(p. 2247\)](#)

## Create and refine an Amazon Transcribe custom vocabulary using an AWS SDK

The following code example shows how to:

- Upload an audio file to Amazon S3.
- Run an Amazon Transcribe job to transcribe the file and get the results.
- Create and refine a custom vocabulary to improve transcription accuracy.
- Run jobs with custom vocabularies and get the results.

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Transcribe an audio file that contains a reading of Jabberwocky by Lewis Carroll. Start by creating functions that wrap Amazon Transcribe actions.

```
def start_job(
    job_name, media_uri, media_format, language_code, transcribe_client,
    vocabulary_name=None):
    """
    Starts a transcription job. This function returns as soon as the job is
    started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.

    :param job_name: The name of the transcription job. This must be unique for
        your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
        in an Amazon S3 bucket.
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
        For example, en-US or ja-JP
    :param transcribe_client: The Boto3 Transcribe client.
    :param vocabulary_name: The name of a custom vocabulary to use when
        transcribing
        the audio file.
    :return: Data about the job.
    """
    try:
        job_args = {
            'TranscriptionJobName': job_name,
            'Media': {'MediaFileUri': media_uri},
            'MediaFormat': media_format,
            'LanguageCode': language_code}
        if vocabulary_name is not None:
            job_args['Settings'] = {'VocabularyName': vocabulary_name}
        response = transcribe_client.start_transcription_job(**job_args)
        job = response['TranscriptionJob']
        logger.info("Started transcription job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't start transcription job %s.", job_name)
```

```

        raise
    else:
        return job

def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The retrieved transcription job.
    """
    try:
        response = transcribe_client.get_transcription_job(
            TranscriptionJobName=job_name)
        job = response['TranscriptionJob']
        logger.info("Got job %s.", job['TranscriptionJobName'])
    except ClientError:
        logger.exception("Couldn't get job %s.", job_name)
        raise
    else:
        return job

def delete_job(job_name, transcribe_client):
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.

    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_transcription_job(
            TranscriptionJobName=job_name)
        logger.info("Deleted job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't delete job %s.", job_name)
        raise

def create_vocabulary(
    vocabulary_name, language_code, transcribe_client,
    phrases=None, table_uri=None):
    """
    Creates a custom vocabulary that can be used to improve the accuracy of
    transcription jobs. This function returns as soon as the vocabulary processing
    is started. Call get_vocabulary to get the current status of the vocabulary.
    The vocabulary is ready to use when its status is 'READY'.

    :param vocabulary_name: The name of the custom vocabulary.
    :param language_code: The language code of the vocabulary.
        For example, en-US or nl-NL.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in the
        vocabulary.
    :return: Information about the newly created vocabulary.
    """
    try:
        vocab_args = {'VocabularyName': vocabulary_name, 'LanguageCode':
language_code}
        if phrases is not None:
            vocab_args['Phrases'] = phrases
        elif table_uri is not None:
            vocab_args['VocabularyFileUri'] = table_uri
        response = transcribe_client.create_vocabulary(**vocab_args)
        logger.info("Created custom vocabulary %s.", response['VocabularyName'])
    except ClientError:
        logger.exception("Couldn't create vocabulary %s.", vocabulary_name)
        raise

```

```

        except ClientError:
            logger.exception("Couldn't create custom vocabulary %s.", vocabulary_name)
            raise
        else:
            return response

    def get_vocabulary(vocabulary_name, transcribe_client):
        """
        Gets information about a custom vocabulary.

        :param vocabulary_name: The name of the vocabulary to retrieve.
        :param transcribe_client: The Boto3 Transcribe client.
        :return: Information about the vocabulary.
        """
        try:
            response = transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
            logger.info("Got vocabulary %s.", response['VocabularyName'])
        except ClientError:
            logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
            raise
        else:
            return response

    def update_vocabulary(
        vocabulary_name, language_code, transcribe_client, phrases=None,
        table_uri=None):
        """
        Updates an existing custom vocabulary. The entire vocabulary is replaced with
        the contents of the update.

        :param vocabulary_name: The name of the vocabulary to update.
        :param language_code: The language code of the vocabulary.
        :param transcribe_client: The Boto3 Transcribe client.
        :param phrases: A list of comma-separated phrases to include in the vocabulary.
        :param table_uri: A table of phrases and pronunciation hints to include in the
                          vocabulary.
        """
        try:
            vocab_args = {'VocabularyName': vocabulary_name, 'LanguageCode':
language_code}
            if phrases is not None:
                vocab_args['Phrases'] = phrases
            elif table_uri is not None:
                vocab_args['VocabularyFileUri'] = table_uri
            response = transcribe_client.update_vocabulary(**vocab_args)
            logger.info(
                "Updated custom vocabulary %s.", response['VocabularyName'])
        except ClientError:
            logger.exception("Couldn't update custom vocabulary %s.", vocabulary_name)
            raise

    def list_vocabularies(vocabulary_filter, transcribe_client):
        """
        Lists the custom vocabularies created for this AWS account.

        :param vocabulary_filter: The returned vocabularies must contain this string in
                                 their names.
        :param transcribe_client: The Boto3 Transcribe client.
        :return: The list of retrieved vocabularies.
        """
        try:
            response = transcribe_client.list_vocabularies(
                NameContains=vocabulary_filter)
            vocabs = response['Vocabularies']
            next_token = response.get('NextToken')
            while next_token is not None:

```

```
        response = transcribe_client.list_vocabularies(
            NameContains=vocabulary_filter, NextToken=next_token)
        vocabs += response['Vocabularies']
        next_token = response.get('NextToken')
    logger.info(
        "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter)
except ClientError:
    logger.exception(
        "Couldn't list vocabularies with filter %s.", vocabulary_filter)
    raise
else:
    return vocabs

def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

Call the wrapper functions to transcribe audio without a custom vocabulary and then with different versions of a custom vocabulary to see improved results.

```
def usage_demo():
    """Shows how to use the Amazon Transcribe service."""
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    s3_resource = boto3.resource('s3')
    transcribe_client = boto3.client('transcribe')

    print('*'*88)
    print("Welcome to the Amazon Transcribe demo!")
    print('*'*88)

    bucket_name = f'jabber-bucket-{time.time_ns()}''
    print(f"Creating bucket {bucket_name}.")
    bucket = s3_resource.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            'LocationConstraint': transcribe_client.meta.region_name})
    media_file_name = '.media/Jabberwocky.mp3'
    media_object_key = 'Jabberwocky.mp3'
    print(f"Uploading media file {media_file_name}.")
    bucket.upload_file(media_file_name, media_object_key)
    media_uri = f's3://{bucket.name}/{media_object_key}'

    job_name_simple = f'Jabber-{time.time_ns()}'
    print(f"Starting transcription job {job_name_simple}.")
    start_job(
        job_name_simple, f's3://{bucket.name}/{media_object_key}', 'mp3', 'en-US',
        transcribe_client)
    transcribe_waiter = TranscribeCompleteWaiter(transcribe_client)
    transcribe_waiter.wait(job_name_simple)
    job_simple = get_job(job_name_simple, transcribe_client)
    transcript_simple = requests.get(
        job_simple['Transcript']['TranscriptFileUri']).json()
```

```

print(f"Transcript for job {transcript_simple['jobName']}:")
print(transcript_simple['results']['transcripts'][0]['transcript'])

print('*88)
print("Creating a custom vocabulary that lists the nonsense words to try to "
      "improve the transcription.")
vocabulary_name = f'Jabber-vocabulary-{time.time_ns()}''
create_vocabulary(
    vocabulary_name, 'en-US', transcribe_client,
    phrases=[

        'brillig', 'slithy', 'borogoves', 'mome', 'raths', 'Jub-Jub',
        'frumious',
        'manxome', 'Tumtum', 'uffish', 'whiffing', 'tulgey', 'thou',
        'frabjous',
        'callooh', 'callay', 'chortled'],
    )
vocabulary_ready_waiter = VocabularyReadyWaiter(transcribe_client)
vocabulary_ready_waiter.wait(vocabulary_name)

job_name_vocabulary_list = f'Jabber-vocabulary-list-{time.time_ns()}''
print(f"Starting transcription job {job_name_vocabulary_list}.")
start_job(
    job_name_vocabulary_list, media_uri, 'mp3', 'en-US', transcribe_client,
    vocabulary_name)
transcribe_waiter.wait(job_name_vocabulary_list)
job_vocabulary_list = get_job(job_name_vocabulary_list, transcribe_client)
transcript_vocabulary_list = requests.get(
    job_vocabulary_list['Transcript']['TranscriptFileUri']).json()
print(f"Transcript for job {transcript_vocabulary_list['jobName']}:")
print(transcript_vocabulary_list['results']['transcripts'][0]['transcript'])

print('*88)
print("Updating the custom vocabulary with table data that provides additional
"
      "pronunciation hints.")
table_vocab_file = 'jabber-vocabulary-table.txt'
bucket.upload_file(table_vocab_file, table_vocab_file)
update_vocabulary(
    vocabulary_name, 'en-US', transcribe_client,
    table_uri=f's3://{bucket.name}/{table_vocab_file}')
vocabulary_ready_waiter.wait(vocabulary_name)

job_name_vocab_table = f'Jabber-vocab-table-{time.time_ns()}''
print(f"Starting transcription job {job_name_vocab_table}.")
start_job(
    job_name_vocab_table, media_uri, 'mp3', 'en-US', transcribe_client,
    vocabulary_name=vocabulary_name)
transcribe_waiter.wait(job_name_vocab_table)
job_vocab_table = get_job(job_name_vocab_table, transcribe_client)
transcript_vocab_table = requests.get(
    job_vocab_table['Transcript']['TranscriptFileUri']).json()
print(f"Transcript for job {transcript_vocab_table['jobName']}:")
print(transcript_vocab_table['results']['transcripts'][0]['transcript'])

print('*88)
print("Getting data for jobs and vocabularies.")
jabber_jobs = list_jobs('Jabber', transcribe_client)
print(f"Found {len(jabber_jobs)} jobs:")
for job_sum in jabber_jobs:
    job = get_job(job_sum['TranscriptionJobName'], transcribe_client)
    print(f"\t{job['TranscriptionJobName']}, {job['Media']['MediaFileUri']}, "
          f"{job['Settings'].get('VocabularyName')}")

jabber_vocabbs = list_vocabularies('Jabber', transcribe_client)
print(f"Found {len(jabber_vocabbs)} vocabularies:")
for vocab_sum in jabber_vocabbs:

```

```
vocab = get_vocabulary(vocab_sum['VocabularyName'], transcribe_client)
vocab_content = requests.get(vocab['DownloadUri']).text
print(f"\t{vocab['VocabularyName']} contents:")
print(vocab_content)

print('*'*88)
print("Deleting demo jobs.")
for job_name in [job_name_simple, job_name_vocabulary_list,
    job_name_vocab_table]:
    delete_job(job_name, transcribe_client)
print("Deleting demo vocabulary.")
delete_vocabulary(vocabulary_name, transcribe_client)
print("Deleting demo bucket.")
bucket.objects.delete()
bucket.delete()
print("Thanks for watching!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateVocabulary](#)
  - [DeleteTranscriptionJob](#)
  - [DeleteVocabulary](#)
  - [GetTranscriptionJob](#)
  - [GetVocabulary](#)
  - [ListVocabularies](#)
  - [StartTranscriptionJob](#)
  - [UpdateVocabulary](#)

## Transcribe audio and get job data with Amazon Transcribe using an AWS SDK

The following code example shows how to:

- Start a transcription job with Amazon Transcribe.
- Wait for the job to complete.
- Get the URI where the transcript is stored.

For more information, see [Getting started with Amazon Transcribe](#).

Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import time
import boto3

def transcribe_file(job_name, file_uri, transcribe_client):
    transcribe_client.start_transcription_job(
        TranscriptionJobName=job_name,
        Media={'MediaFileUri': file_uri},
```

```
        MediaFormat='wav',
        LanguageCode='en-US'
    )

    max_tries = 60
    while max_tries > 0:
        max_tries -= 1
        job =
transcribe_client.get_transcription_job(TranscriptionJobName=job_name)
        job_status = job['TranscriptionJob']['TranscriptionJobStatus']
        if job_status in ['COMPLETED', 'FAILED']:
            print(f"Job {job_name} is {job_status}.")
            if job_status == 'COMPLETED':
                print(
                    f"Download the transcript from\n"
                    f"\t{job['TranscriptionJob']['Transcript']}"
                    ['TranscriptFileUri']).")
                break
            else:
                print(f"Waiting for {job_name}. Current status is {job_status}.")
                time.sleep(10)

def main():
    transcribe_client = boto3.client('transcribe')
    file_uri = 's3://test-transcribe/answer2.wav'
    transcribe_file('Example-job', file_uri, transcribe_client)

if __name__ == '__main__':
    main()
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [GetTranscriptionJob](#)
  - [StartTranscriptionJob](#)

## Cross-service examples for Amazon Transcribe using AWS SDKs

The following code examples show how to use Amazon Transcribe with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Build an Amazon Transcribe app \(p. 2248\)](#)
- [Build an Amazon Transcribe streaming app \(p. 2249\)](#)
- [Convert text to speech and back to text using an AWS SDK \(p. 2249\)](#)

## Build an Amazon Transcribe app

The following code example shows how to use Amazon Transcribe to transcribe and display voice recordings in the browser.

JavaScript

### SDK for JavaScript V3

Create an app that uses Amazon Transcribe to transcribe and display voice recordings in the browser. The app uses two Amazon Simple Storage Service (Amazon S3) buckets, one to host the application code, and another to store transcriptions. The app uses an Amazon Cognito user pool to authenticate your users. Authenticated users have AWS Identity and Access Management (IAM) permissions to access the required AWS services.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

#### Services used in this example

- Amazon Cognito Identity
- Amazon S3
- Amazon Transcribe

## Build an Amazon Transcribe streaming app

The following code example shows how to build an app that records, transcribes, and translates live audio in real-time, and emails the results.

JavaScript

### SDK for JavaScript V3

Shows how to use Amazon Transcribe to build an app that records, transcribes, and translates live audio in real-time, and emails the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

## Convert text to speech and back to text using an AWS SDK

The following code example shows how to:

- Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file.
- Upload the audio file to an Amazon Simple Storage Service (Amazon S3) bucket.
- Use Amazon Transcribe to convert the audio file to text.
- Display the text.

Rust

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file, upload the audio file to an Amazon Simple Storage Service bucket, use Amazon Transcribe to convert that audio file to text, and display the text.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Polly
- Amazon S3
- Amazon Transcribe

## Code examples for Amazon Translate using AWS SDKs

The following code examples show how to use Amazon Translate with an AWS software development kit (SDK).

### Code examples

- [Actions for Amazon Translate using AWS SDKs \(p. 2250\)](#)
  - [Describe an Amazon Translate translation job using an AWS SDK \(p. 2251\)](#)
  - [List the Amazon Translate translation jobs using an AWS SDK \(p. 2253\)](#)
  - [Start an Amazon Translate translation job using an AWS SDK \(p. 2255\)](#)
  - [Stop an Amazon Translate translation job using an AWS SDK \(p. 2257\)](#)
  - [Translate text with Amazon Translate using an AWS SDK \(p. 2259\)](#)
- [Scenarios for Amazon Translate using AWS SDKs \(p. 2261\)](#)
  - [Get started with Amazon Translate jobs using an AWS SDK \(p. 2261\)](#)
- [Cross-service examples for Amazon Translate using AWS SDKs \(p. 2263\)](#)
  - [Build an Amazon Transcribe streaming app \(p. 2263\)](#)
  - [Create an Amazon Lex Chatbot within a web application to engage your web site visitors \(p. 2264\)](#)
  - [Build a publish and subscription application that translates messages \(p. 2265\)](#)

## Actions for Amazon Translate using AWS SDKs

The following code examples show how to use Amazon Translate with AWS SDKs. Each example calls an individual service function.

### Examples

- [Describe an Amazon Translate translation job using an AWS SDK \(p. 2251\)](#)

- [List the Amazon Translate translation jobs using an AWS SDK \(p. 2253\)](#)
- [Start an Amazon Translate translation job using an AWS SDK \(p. 2255\)](#)
- [Stop an Amazon Translate translation job using an AWS SDK \(p. 2257\)](#)
- [Translate text with Amazon Translate using an AWS SDK \(p. 2259\)](#)

## Describe an Amazon Translate translation job using an AWS SDK

The following code examples show how to describe an Amazon Translate translation job.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

public class DescribeTextTranslation
{
    public static async Task Main()
    {
        var client = new AmazonTranslateClient();

        // The Job Id is generated when the text translation job is started
        // with a call to the StartTextTranslationJob method.
        var jobId = "1234567890abcdef01234567890abcde";

        var request = new DescribeTextTranslationJobRequest
        {
            JobId = jobId,
        };

        var jobProperties = await DescribeTranslationJobAsync(client, request);

        DisplayTranslationJobDetails(jobProperties);
    }

    ///<summary>
    /// Retrieve information about an Amazon Translate text translation job.
    ///</summary>
    ///<param name="client">The initialized Amazon Translate client object.</param>
    ///<param name="request">The DescribeTextTranslationJobRequest object.</param>
    ///<returns>The TextTranslationJobProperties object containing
    /// information about the text translation job..</returns>
    public static async Task<TextTranslationJobProperties>
DescribeTranslationJobAsync(
    AmazonTranslateClient client,
    DescribeTextTranslationJobRequest request)
{
    var response = await client.DescribeTextTranslationJobAsync(request);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        return response.TextTranslationJobProperties;
    }
}
```

```
        }
    else
    {
        return null;
    }
}

/// <summary>
/// Displays the properties of the text translation job.
/// </summary>
/// <param name="jobProperties">The properties of the text translation
/// job returned by the call to DescribeTextTranslationJobAsync.</param>
public static void
DisplayTranslationJobDetails(TextTranslationJobProperties jobProperties)
{
    if (jobProperties is null)
    {
        Console.WriteLine("No text translation job properties found.");
        return;
    }

    // Display the details of the text translation job.
    Console.WriteLine($"{jobProperties.JobId}: {jobProperties.JobName}");
}
```

- For API details, see [DescribeTextTranslationJob](#) in *AWS SDK for .NET API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Gets the properties associated with an asynchronous batch translation job"
"including name, ID, status, source and target languages, input/output S3
buckets, and so on."
TRY.
    oo_result = lo_xl8->describetexttranslationjob(          "oo_result is returned
for testing purpose"
        EXPORTING
            iv_jobid      = iv_jobid
        ).
    MESSAGE 'Job description retrieved' TYPE 'I'.
    CATCH /aws1/cx_xl8internalserverex .
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
    CATCH /aws1/cx_xl8resourcenotfoundex .
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
    CATCH /aws1/cx_xl8toomanyrequestsex.
        MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
ENDTRY.
```

- For API details, see [DescribeTextTranslationJob](#) in *AWS SDK for SAP ABAP API reference*.

## List the Amazon Translate translation jobs using an AWS SDK

The following code examples show how to list the Amazon Translate translation jobs.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

public class ListTranslationJobs
{
    public static async Task Main()
    {
        var client = new AmazonTranslateClient();
        var filter = new TextTranslationJobFilter
        {
            JobStatus = "COMPLETED",
        };

        var request = new ListTextTranslationJobsRequest
        {
            MaxResults = 10,
            Filter = filter,
        };

        await ListJobsAsync(client, request);
    }

    ///<summary>
    /// List Amazon Translate text translation jobs.
    ///</summary>
    ///<param name="client">The initialized Amazon Translate client object.</param>
    ///<param name="request">An Amazon Translate
    /// ListTextTranslationJobsRequest object detailing which text
    /// translation jobs are of interest.</param>
    public static async Task ListJobsAsync(
        AmazonTranslateClient client,
        ListTextTranslationJobsRequest request)
    {
        ListTextTranslationJobsResponse response;

        do
        {
            response = await client.ListTextTranslationJobsAsync(request);

            ShowTranslationJobDetails(response.TextTranslationJobPropertiesList);

            request.NextToken = response.NextToken;
        }
        while (response.NextToken is not null);
    }
}
```

```
        }

        ///<summary>
        /// List existing translation job details.
        /// </summary>
        ///<param name="properties">A list of Amazon Translate text
        /// translation jobs.</param>
        public static void
ShowTranslationJobDetails(List<TextTranslationJobProperties> properties)
{
    properties.ForEach(prop =>
    {
        Console.WriteLine($"{{prop.JobId}}: {{prop.JobName}}");
        Console.WriteLine($"Status: {{prop.JobStatus}}");
        Console.WriteLine($"Submitted time: {{prop.SubmittedTime}}");
    });
}
```

- For API details, see [ListTextTranslationJobs](#) in *AWS SDK for .NET API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Gets a list of the batch translation jobs that you have submitted."

DATA lo_filter TYPE REF TO /aws1/cl_xl8textxlationjobfilt.

"Create an ABAP object for filtering using jobname"
CREATE OBJECT lo_filter
    EXPORTING
        iv_jobname = iv_jobname.

TRY.
    oo_result = lo_xl8->listtexttranslationjobs(          "oo_result is returned
for testing purpose"
        EXPORTING
            io_filter      = lo_filter
        ).
MESSAGE 'Jobs retrieved' TYPE 'I'.
CATCH /aws1/cx_xl8internalserverex .
    MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
CATCH /aws1/cx_xl8invalidfilterex .
    MESSAGE 'The filter specified for the operation is not valid. Specify a
different filter.' TYPE 'E'.
CATCH /aws1/cx_xl8invalidrequestex .
    MESSAGE 'The request that you made is not valid.' TYPE 'E'.
CATCH /aws1/cx_xl8toomanyrequestsex.
    MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
ENDTRY.
```

- For API details, see [ListTextTranslationJobs](#) in *AWS SDK for SAP ABAP API reference*.

## Start an Amazon Translate translation job using an AWS SDK

The following code examples show how to start an Amazon Translate translation job.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

public class BatchTranslate
{
    public static async Task Main()
    {
        var contentType = "text/plain";

        // Set this variable to an S3 bucket location with a folder.
        // Input files must be in a folder and not at the bucket root.
        var s3InputUri = "s3://DOC-EXAMPLE-BUCKET1/FOLDER/";
        var s3OutputUri = "s3://DOC-EXAMPLE-BUCKET2/";

        // This role must have permissions to read the source bucket and to
        // write to the destination bucket where the translated text will be
        // stored.
        var dataAccessRoleArn = "arn:aws:iam::0123456789ab:role/
S3TranslateRole";

        var client = new AmazonTranslateClient();

        var inputConfig = new InputDataConfig
        {
            ContentType = contentType,
            S3Uri = s3InputUri,
        };

        var outputConfig = new OutputDataConfig
        {
            S3Uri = s3OutputUri,
        };

        var request = new StartTextTranslationJobRequest
        {
            JobName = "ExampleTranslationJob",
            DataAccessRoleArn = dataAccessRoleArn,
            InputDataConfig = inputConfig,
            OutputDataConfig = outputConfig,
            SourceLanguageCode = "en",
            TargetLanguageCodes = new List<string> { "fr" },
        };
    }
}
```

```
        var response = await StartTextTranslationAsync(client, request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"{response.JobId}: {response.JobStatus}");
        }
    }

    ///<summary>
    ///<summary>Start the Amazon Translate text translation job.
    ///</summary>
    ///<param name="client">The initialized AmazonTranslateClient object.</param>
    ///<param name="request">The request object that includes details such
    ///as source and destination bucket names and the IAM Role that will
    ///be used to access the buckets.</param>
    ///<returns>The StartTextTranslationResponse object that includes the
    ///details of the request response.</returns>
    public static async Task<StartTextTranslationJobResponse>
StartTextTranslationAsync(AmazonTranslateClient client,
StartTextTranslationJobRequest request)
{
    var response = await client.StartTextTranslationJobAsync(request);
    return response;
}
}
```

- For API details, see [StartTextTranslationJob](#) in *AWS SDK for .NET API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Starts an asynchronous batch translation job."
"Use batch translation jobs to translate large volumes of text across multiple
documents at once."

DATA lo_inputdataconfig  TYPE REF TO /aws1/cl_xl8inputdataconfig.
DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

"Create an ABAP object for the input data config"
CREATE OBJECT lo_inputdataconfig
    EXPORTING
        iv_s3uri      = iv_input_data_s3uri
        iv_contenttype = iv_input_data_contenttype.

"Create an ABAP object for the output data config"
CREATE OBJECT lo_outputdataconfig
```

```
EXPORTING
    iv_s3uri = iv_output_data_s3uri.

"Create an interal table for target languages"
CREATE OBJECT lo_targetlanguagecodes
    EXPORTING
        iv_value = iv_targetlanguagecode.
INSERT lo_targetlanguagecodes INTO TABLE lt_targetlanguagecodes.

TRY.
    oo_result = lo_xl8->starttexttranslationjob(          "oo_result is returned
for testing purpose"
        EXPORTING
            io_inputdataconfig = lo_inputdataconfig
            io_outputdataconfig = lo_outputdataconfig
            it_targetlanguagecodes = lt_targetlanguagecodes
            iv_dataaccessrolearn = iv_dataaccessrolearn
            iv_jobname = iv_jobname
            iv_sourcelanguagecode = iv_sourcelanguagecode
        ).
    MESSAGE 'Translation job started' TYPE 'I'.
    CATCH /aws1/cx_xl8internalserverex .
        MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
    CATCH /aws1/cx_xl8invparamvalueex .
        MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
    CATCH /aws1/cx_xl8invalidrequestex .
        MESSAGE 'The request that you made is not valid.' TYPE 'E'.
    CATCH /aws1/cx_xl8resourcenotfoundex .
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
    CATCH /aws1/cx_xl8toomanyrequestsex .
        MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
    CATCH /aws1/cx_xl8unsuppedlanguage00 .
        MESSAGE 'Amazon Translate does not support translation from the language of
the source text into the requested target language.' TYPE 'E'.
ENDTRY.
```

- For API details, see [StartTextTranslationJob](#) in *AWS SDK for SAP ABAP API reference*.

## Stop an Amazon Translate translation job using an AWS SDK

The following code examples show how to stop an Amazon Translate translation job.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

class StopTextTranslationJob
{
    public static async Task Main()
    {
        var client = new AmazonTranslateClient();
```

```
        var jobId = "1234567890abcdef01234567890abcde";

        var request = new StopTextTranslationJobRequest
        {
            JobId = jobId,
        };

        await StopTranslationJobAsync(client, request);
    }

    ///<summary>
    /// Sends a request to stop a text translation job.
    ///</summary>
    ///<param name="client">Initialized AmazonTrnslateClient object.</param>
    ///<param name="request">The request object to be passed to the
    /// StopTextJobAsync method.</param>
    public static async Task StopTranslationJobAsync(
        AmazonTranslateClient client,
        StopTextTranslationJobRequest request)
    {
        var response = await client.StopTextTranslationJobAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"{response.JobId} as status:
{response.JobStatus}");
        }
    }
}
```

- For API details, see [StopTextTranslationJob](#) in [AWS SDK for .NET API Reference](#).

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Stops an asynchronous batch translation job that is in progress."

TRY.
    oo_result = lo_xl8->stoptexttranslationjob(      "oo_result is returned for
testing purpose"
        EXPORTING
            iv_jobid      = iv_jobid
        ).
    MESSAGE 'Translation job stopped' TYPE 'I'.
CATCH /aws1/cx_xl8internalserverex .
    MESSAGE 'An internal server error occurred.' TYPE 'E'.
CATCH /aws1/cx_xl8resourcenotfoundex .
    MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
CATCH /aws1/cx_xl8toomanyrequestsx.
    MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
ENDTRY.
```

- For API details, see [StopTextTranslationJob](#) in *AWS SDK for SAP ABAP API reference*.

## Translate text with Amazon Translate using an AWS SDK

The following code examples show how to translate text with Amazon Translate.

.NET

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;
using Amazon.Translate;
using Amazon.Translate.Model;

public class TranslateText
{
    public static async Task Main()
    {
        // If the region you want to use is different from the region
        // defined for the default user, supply it as a parameter to the
        // Amazon Translate client object constructor.
        var client = new AmazonTranslateClient();

        // Set the source language to "auto" to request Amazon Translate to
        // automatically detect the language of the source text.

        // You can get a list of the languages supported by Amazon Translate
        // in the Amazon Translate Developer's Guide here:
        //     https://docs.aws.amazon.com/translate/latest/dg/what-is.html
        string srcLang = "en"; // English.
        string destLang = "fr"; // French.

        // The Amazon Simple Storage Service (Amazon S3) bucket where the
        // source text file is stored.
        string srcBucket = "DOC-EXAMPLE-BUCKET";
        string srcTextFile = "source.txt";

        var srcText = await GetSourceTextAsync(srcBucket, srcTextFile);
        var destText = await TranslatingTextAsync(client, srcLang, destLang,
srcText);

        ShowText(srcText, destText);
    }

    /// <summary>
    /// Use the Amazon S3 TransferUtility to retrieve the text to translate
    /// from an object in an S3 bucket.
    /// </summary>
    /// <param name="srcBucket">The name of the S3 bucket where the
    /// text is stored.
    /// </param>
```

```

    ///<param name="srcTextFile">The key of the S3 object that
    ///contains the text to translate.</param>
    ///<returns>A string representing the source text.</returns>
    public static async Task<string> GetSourceTextAsync(string srcBucket,
string srcTextFile)
{
    string srcText = string.Empty;

    var s3Client = new AmazonS3Client();
    TransferUtility utility = new TransferUtility(s3Client);

    using var stream = await utility.OpenStreamAsync(srcBucket,
srcTextFile);

    StreamReader file = new System.IO.StreamReader(stream);

    srcText = file.ReadToEnd();
    return srcText;
}

///<summary>
/// Use the Amazon Translate Service to translate the document from the
/// source language to the specified destination language.
///</summary>
///<param name="client">The Amazon Translate Service client used to
/// perform the translation.</param>
///<param name="srcLang">The language of the source text.</param>
///<param name="destLang">The destination language for the translated
/// text.</param>
///<param name="text">A string representing the text to translate.</param>
///<returns>The text that has been translated to the destination
/// language.</returns>
public static async Task<string> TranslatingTextAsync(AmazonTranslateClient
client, string srcLang, string destLang, string text)
{
    var request = new TranslateTextRequest
    {
        SourceLanguageCode = srcLang,
        TargetLanguageCode = destLang,
        Text = text,
    };

    var response = await client.TranslateTextAsync(request);

    return response.TranslatedText;
}

///<summary>
/// Show the original text followed by the translated text.
///</summary>
///<param name="srcText">The original text to be translated.</param>
///<param name="destText">The translated text.</param>
public static void ShowText(string srcText, string destText)
{
    Console.WriteLine("Source text:");
    Console.WriteLine(srcText);
    Console.WriteLine();
    Console.WriteLine("Translated text:");
    Console.WriteLine(destText);
}
}

```

- For API details, see [TranslateText](#) in *AWS SDK for .NET API Reference*.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Translates input text from the source language to the target language."
TRY.
    oo_result = lo_xl8->translatetext(      "oo_result is returned for testing
purpose"
    EXPORTING
        iv_text          = iv_text
        iv_sourcelanguagecode = iv_sourcelanguagecode
        iv_targetlanguagecode = iv_targetlanguagecode
    ).
    MESSAGE 'Translation completed' TYPE 'I'.
    CATCH /aws1/cx_xl8detectedlanguage00 .
        MESSAGE 'The confidence that Amazon Comprehend accurately detected the
source language is low.' TYPE 'E'.
    CATCH /aws1/cx_xl8internalserverex .
        MESSAGE 'An internal server error occurred.' TYPE 'E'.
    CATCH /aws1/cx_xl8invalidrequestex .
        MESSAGE 'The request that you made is not valid.' TYPE 'E'.
    CATCH /aws1/cx_xl8resourcenotfoundex .
        MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
    CATCH /aws1/cx_xl8serviceunavailex .
        MESSAGE 'The Amazon Translate service is temporarily unavailable.' TYPE
'E'.
    CATCH /aws1/cx_xl8textsizeilmtexdex .
        MESSAGE 'The size of the text you submitted exceeds the size limit. ' TYPE
'E'.
    CATCH /aws1/cx_xl8toomanyrequestsex .
        MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
    CATCH /aws1/cx_xl8unsuppedlanguage00 .
        MESSAGE 'Amazon Translate does not support translation from the language of
the source text into the requested target language. ' TYPE 'E'.
ENDTRY.
```

- For API details, see [TranslateText](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios for Amazon Translate using AWS SDKs

The following code examples show how to use Amazon Translate with AWS SDKs. Each example shows you how to accomplish a specific task by calling multiple functions within the same service.

### Examples

- [Get started with Amazon Translate jobs using an AWS SDK \(p. 2261\)](#)

## Get started with Amazon Translate jobs using an AWS SDK

The following code example shows how to:

- Start an asynchronous batch translation job.
- Wait for the asynchronous job to complete.
- Describe the asynchronous job.

## SAP ABAP

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_inputdataconfig TYPE REF TO /aws1/cl_xl8inputdataconfig.
DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

"Create an ABAP object for the input data config"
CREATE OBJECT lo_inputdataconfig
  EXPORTING
    iv_s3uri      = iv_input_data_s3uri
    iv_contenttype = iv_input_data_contenttype.

"Create an ABAP object for the output data config"
CREATE OBJECT lo_outputdataconfig
  EXPORTING
    iv_s3uri = iv_output_data_s3uri.

"Create an interal table for target languages"
CREATE OBJECT lo_targetlanguagecodes
  EXPORTING
    iv_value = iv_targetlanguagecode.
INSERT lo_targetlanguagecodes INTO TABLE lt_targetlanguagecodes.

TRY.
  DATA(lo_translationjob_result) = lo_xl8->starttexttranslationjob(
    EXPORTING
      io_inputdataconfig = lo_inputdataconfig
      io_outputdataconfig = lo_outputdataconfig
      it_targetlanguagecodes = lt_targetlanguagecodes
      iv_dataaccessrolearn = iv_dataaccessrolearn
      iv_jobname = iv_jobname
      iv_sourcelanguagecode = iv_sourcelanguagecode
    ).
  MESSAGE 'Translation job started' TYPE 'I'.
  CATCH /aws1/cx_xl8internalserverex .
    MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
  CATCH /aws1/cx_xl8invparamvalueex .
    MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
  CATCH /aws1/cx_xl8invalidrequestex.
    MESSAGE 'The request that you made is not valid.' TYPE 'E'.
  CATCH /aws1/cx_xl8resourcenotfoundex .
    MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
  CATCH /aws1/cx_xl8toomanyrequestsex.
    MESSAGE 'You have made too many requests within a short period of time. '
TYPE 'E'.
  CATCH /aws1/cx_xl8unsuppedlanguage00 .
```

```
MESSAGE 'Amazon Translate does not support translation from the language of
the source text into the requested target language.' TYPE 'E'.
ENDTRY.

"Get the job ID"
DATA(lv_jobid) = lo_translationjob_result->get_jobid( ).

"Wait for translate job to complete"
DATA(lo_des_translation_result) = lo_xl8->describetexttranslationjob( iv_jobid
= lv_jobid ).
WHILE lo_des_translation_result->get_textxlationjobproperties( )-
>get_jobstatus( ) <> 'COMPLETED'.
    IF sy-index = 30.
        EXIT.                      "maximum 900 seconds"
    ENDIF.
    WAIT UP TO 30 SECONDS.
    lo_des_translation_result = lo_xl8->describetexttranslationjob( iv_jobid =
lv_jobid ).
ENDWHILE.

TRY.
    oo_result = lo_xl8->describetexttranslationjob(           "oo_result is returned
for testing purpose"
        EXPORTING
            iv_jobid      = lv_jobid
        ).
MESSAGE 'Job description retrieved' TYPE 'I'.
CATCH /aws1/cx_xl8internalserverex .
    MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
CATCH /aws1/cx_xl8resourcenotfoundex .
    MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
CATCH /aws1/cx_xl8toomanyrequestsex.
    MESSAGE 'You have made too many requests within a short period of time.'
TYPE 'E'.
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [DescribeTextTranslationJob](#)
  - [StartTextTranslationJob](#)

## Cross-service examples for Amazon Translate using AWS SDKs

The following code examples show how to use Amazon Translate with AWS SDKs. Each example contains a sample application that works across multiple AWS services.

### Examples

- [Build an Amazon Transcribe streaming app \(p. 2263\)](#)
- [Create an Amazon Lex Chatbot within a web application to engage your web site visitors \(p. 2264\)](#)
- [Build a publish and subscription application that translates messages \(p. 2265\)](#)

## Build an Amazon Transcribe streaming app

The following code example shows how to build an app that records, transcribes, and translates live audio in real-time, and emails the results.

JavaScript

### SDK for JavaScript V3

Shows how to use Amazon Transcribe to build an app that records, transcribes, and translates live audio in real-time, and emails the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

## Create an Amazon Lex Chatbot within a web application to engage your web site visitors

The following code examples show how to create a Chatbot to engage your web site visitors.

Java

### SDK for Java 2.x

Shows how to use the Amazon Lex API to create a Chatbot within a web application to engage your web site visitors.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

JavaScript

### SDK for JavaScript V3

Shows how to use the Amazon Lex API to create a Chatbot within a web application to engage your web site visitors.

For complete source code and instructions on how to set up and run, see the full example [Building an Amazon Lex chatbot](#) in the AWS SDK for JavaScript developer guide.

#### Services used in this example

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

## Build a publish and subscription application that translates messages

The following code examples show how to create an application that has subscription and publish functionality and translates messages.

.NET

### AWS SDK for .NET

Shows how to use the Amazon Simple Notification Service .NET API to create a web application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon SNS
- Amazon Translate

Java

### SDK for Java 2.x

Shows how to use the Amazon Simple Notification Service Java API to create a web application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

For complete source code and instructions on how to set up and run the example that uses the Java Async API, see the full example on [GitHub](#).

#### Services used in this example

- Amazon SNS
- Amazon Translate

Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to use the Amazon SNS Kotlin API to create an application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to create a web app, see the full example on [GitHub](#).

For complete source code and instructions on how to create a native Android app, see the full example on [GitHub](#).

**Services used in this example**

- Amazon SNS
- Amazon Translate

# Code examples by SDK using AWS SDKs

The following code examples show how to use AWS services with an AWS software development kit (SDK).

## Code examples

- [Code examples for AWS SDK for .NET \(p. 2267\)](#)
- [Code examples for SDK for C++ \(p. 2608\)](#)
- [Code examples for SDK for Go V2 \(p. 2715\)](#)
- [Code examples for SDK for JavaScript V2 \(p. 2801\)](#)
- [Code examples for SDK for JavaScript V3 \(p. 2909\)](#)
- [Code examples for SDK for Java 2.x \(p. 3156\)](#)
- [Code examples for SDK for Kotlin \(p. 3499\)](#)
- [Code examples for SDK for PHP \(p. 3665\)](#)
- [Code examples for SDK for Python \(Boto3\) \(p. 3736\)](#)
- [Code examples for SDK for Ruby \(p. 4310\)](#)
- [Code examples for SDK for Rust \(p. 4391\)](#)
- [Code examples for SDK for SAP ABAP \(p. 4482\)](#)
- [Code examples for SDK for Swift \(p. 4557\)](#)

## Code examples for AWS SDK for .NET

The code examples in this topic show you how to use the AWS SDK for .NET with AWS.

### Examples

- [Single-service actions and scenarios using AWS SDK for .NET \(p. 2267\)](#)
- [Cross-service examples using AWS SDK for .NET \(p. 2606\)](#)

## Single-service actions and scenarios using AWS SDK for .NET

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for .NET with AWS services.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Services

- [ACM examples using AWS SDK for .NET \(p. 2268\)](#)
- [CloudWatch examples using AWS SDK for .NET \(p. 2271\)](#)
- [CloudWatch Logs examples using AWS SDK for .NET \(p. 2280\)](#)
- [Amazon Cognito Identity Provider examples using AWS SDK for .NET \(p. 2287\)](#)
- [Amazon Comprehend examples using AWS SDK for .NET \(p. 2297\)](#)
- [DynamoDB examples using AWS SDK for .NET \(p. 2305\)](#)
- [Amazon EC2 examples using AWS SDK for .NET \(p. 2344\)](#)
- [Amazon EC2 Auto Scaling examples using AWS SDK for .NET \(p. 2355\)](#)
- [AWS Glue examples using AWS SDK for .NET \(p. 2370\)](#)
- [IAM examples using AWS SDK for .NET \(p. 2388\)](#)
- [AWS KMS examples using AWS SDK for .NET \(p. 2409\)](#)
- [Kinesis examples using AWS SDK for .NET \(p. 2416\)](#)
- [Lambda examples using AWS SDK for .NET \(p. 2425\)](#)
- [Organizations examples using AWS SDK for .NET \(p. 2439\)](#)
- [Amazon Polly examples using AWS SDK for .NET \(p. 2450\)](#)
- [Amazon RDS examples using AWS SDK for .NET \(p. 2458\)](#)
- [Amazon Rekognition examples using AWS SDK for .NET \(p. 2479\)](#)
- [Route 53 domain registration examples using AWS SDK for .NET \(p. 2499\)](#)
- [Amazon S3 examples using AWS SDK for .NET \(p. 2516\)](#)
- [S3 Glacier examples using AWS SDK for .NET \(p. 2531\)](#)
- [Amazon SES examples using AWS SDK for .NET \(p. 2538\)](#)
- [Amazon SNS examples using AWS SDK for .NET \(p. 2545\)](#)
- [Amazon SQS examples using AWS SDK for .NET \(p. 2554\)](#)
- [AWS STS examples using AWS SDK for .NET \(p. 2568\)](#)
- [Secrets Manager examples using AWS SDK for .NET \(p. 2570\)](#)
- [AWS Support examples using AWS SDK for .NET \(p. 2572\)](#)
- [Amazon Transcribe examples using AWS SDK for .NET \(p. 2592\)](#)
- [Amazon Translate examples using AWS SDK for .NET \(p. 2599\)](#)

## ACM examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with AWS Certificate Manager.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2268\)](#)

## Actions

### Describe a certificate

The following code example shows how to describe ACM certificates.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon;
using Amazon.CertificateManager;
using Amazon.CertificateManager.Model;
using System;
using System.Threading.Tasks;

namespace DescribeCertificate
{
    class DescribeCertificate
    {
        // The following example retrieves and displays the metadata for a
        // certificate using the AWS Certificate Manager (ACM) service. It
        // was created using AWS SDK for .NET 3.5 and .NET 5.0.

        // Specify your AWS Region (an example Region is shown).
        private static readonly RegionEndpoint ACMRegion = RegionEndpoint.USEast1;
        private static AmazonCertificateManagerClient _client;

        static void Main(string[] args)
        {
            var _client = new
                Amazon.CertificateManager.AmazonCertificateManagerClient(ACMRegion);

            var describeCertificateReq = new DescribeCertificateRequest();
            // The ARN used here is just an example. Replace it with the ARN of
            // a certificate that exists on your account.
            describeCertificateReq.CertificateArn =
                "arn:aws:acm:us-
east-1:123456789012:certificate/8cf7daa-9b6a-2d07-92bc-1c309EXAMPLE";

            var certificateDetailResp =
                DescribeCertificateResponseAsync(client: _client, request:
describeCertificateReq);
            var certificateDetail = certificateDetailResp.Result.Certificate;

            if (certificateDetail is not null)
            {
                DisplayCertificateDetails(certificateDetail);
            }
        }

        ///<summary>
        /// Displays detailed metadata about a certificate retrieved
        /// using the ACM service.
        ///</summary>
        ///<param name="certificateDetail">The object that contains details
        /// returned from the call to DescribeCertificateAsync.</param>
        static void DisplayCertificateDetails(CertificateDetail certificateDetail)
        {
            Console.WriteLine("\nCertificate Details: ");
            Console.WriteLine($"Certificate Domain: {certificateDetail.DomainName}");
            Console.WriteLine($"Certificate Arn: {certificateDetail.CertificateArn}");
            Console.WriteLine($"Certificate Subject: {certificateDetail.Subject}");
            Console.WriteLine($"Certificate Status: {certificateDetail.Status}");
            foreach (var san in certificateDetail.SubjectAlternativeNames)
            {
                Console.WriteLine($"Certificate SubjectAlternativeName: {san}");
            }
        }
    }
}
```

```
}

/// <summary>
/// Retrieves the metadata associated with the ACM service certificate.
/// </summary>
/// <param name="client">An AmazonCertificateManagerClient object
/// used to call DescribeCertificateResponse.</param>
/// <param name="request">The DescribeCertificateRequest object that
/// will be passed to the method call.</param>
/// <returns></returns>
static async Task<DescribeCertificateResponse>
DescribeCertificateResponseAsync(
    AmazonCertificateManagerClient client, DescribeCertificateRequest request)
{
    var response = new DescribeCertificateResponse();

    try
    {
        response = await client.DescribeCertificateAsync(request);
    }
    catch (InvalidArnException ex)
    {
        Console.WriteLine($"Error: The ARN specified is invalid.");
    }
    catch (ResourceNotFoundException ex)
    {
        Console.WriteLine($"Error: The specified certificate could not be
found.");
    }

    return response;
}
}
```

- For API details, see [DescribeCertificate](#) in *AWS SDK for .NET API Reference*.

## List certificates

The following code example shows how to list ACM certificates.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon;
using Amazon.CertificateManager;
using Amazon.CertificateManager.Model;
using System;
using System.Threading.Tasks;

namespace ListCertificates
{
    // The following example retrieves and displays a list of the
    // certificates defined for the default account using the AWS
    // Certificate Manager (ACM) service. It was created using
```

```
// AWS SDK for .NET 3.5 and .NET 5.0.
class ListCertificates
{
    // Specify your AWS Region (an example Region is shown).

    private static readonly RegionEndpoint ACMRegion = RegionEndpoint.USEast1;
    private static AmazonCertificateManagerClient _client;

    static void Main(string[] args)
    {
        var _client = new AmazonCertificateManagerClient(ACMRegion);
        var certificateList = ListCertificatesResponseAsync(client: _client);

        Console.WriteLine("Certificate Summary List\n");

        foreach (var certificate in certificateList.Result.CertificateSummaryList)
        {
            Console.WriteLine($"Certificate Domain: {certificate.DomainName}");
            Console.WriteLine($"Certificate ARN: {certificate.CertificateArn}\n");
        }
    }

    ///<summary>
    ///<summary>Retrieves a list of the certificates defined in this Region.
    ///</summary>
    ///<param name="client">The ACM client object passed to the
    ///<param name="request"></param>
    ///<returns>The ListCertificatesResponse.</returns>
    static async Task<ListCertificatesResponse> ListCertificatesResponseAsync(
        AmazonCertificateManagerClient client)
    {
        var request = new ListCertificatesRequest();

        var response = await client.ListCertificatesAsync(request);
        return response;
    }
}
```

- For API details, see [ListCertificates](#) in *AWS SDK for .NET API Reference*.

## CloudWatch examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon CloudWatch.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2272\)](#)

## Actions

### Delete alarms

The following code example shows how to delete Amazon CloudWatch alarms.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// This example shows how to delete Amazon CloudWatch alarms. The example
/// was created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DeleteAlarms
{
    public static async Task Main()
    {
        IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();

        var alarmNames = CreateAlarmNameList();
        await DeleteAlarmsAsyncExample(cwClient, alarmNames);
    }

    /// <summary>
    /// Delete the alarms whose names are listed in the alarmNames parameter.
    /// </summary>
    /// <param name="client">The initialized Amazon CloudWatch client.</param>
    /// <param name="alarmNames">A list of names for the alarms to be
    /// deleted.</param>
    public static async Task DeleteAlarmsAsyncExample(IAmazonCloudWatch client,
List<string> alarmNames)
    {
        var request = new DeleteAlarmsRequest
        {
            AlarmNames = alarmNames,
        };

        try
        {
            var response = await client.DeleteAlarmsAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine("Alarms successfully deleted:");
                alarmNames
                    .ForEach(name => Console.WriteLine($"{name}"));
            }
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
    }

    /// <summary>
```

```
    /// Defines and returns the list of alarm names to delete.  
    /// </summary>  
    /// <returns>A list of alarm names.</returns>  
    public static List<string> CreateAlarmNameList()  
    {  
        // The list of alarm names passed to DeleteAlarmsAsync  
        // can contain up to 100 alarm names.  
        var theList = new List<string>  
        {  
            "ALARM_NAME_1",  
            "ALARM_NAME_2",  
        };  
  
        return theList;  
    }  
}
```

- For API details, see [DeleteAlarms](#) in *AWS SDK for .NET API Reference*.

## Describe alarm history

The following code example shows how to describe Amazon CloudWatch alarm history.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get a list of CloudWatch alarms, then retrieve the history for each alarm.

```
using System;  
using System.Threading.Tasks;  
using Amazon.CloudWatch;  
using Amazon.CloudWatch.Model;  
  
    /// <summary>  
    /// This example retrieves a list of Amazon CloudWatch alarms and, for  
    /// each one, displays its history. The example was created using the  
    /// AWS SDK for .NET 3.7 and .NET Core 5.0.  
    /// </summary>  
public class DescribeAlarmHistories  
{  
    /// <summary>  
    /// Retrieves a list of alarms and then passes each name to the  
    /// DescribeAlarmHistoriesAsync method to retrieve its history.  
    /// </summary>  
    public static async Task Main()  
    {  
        IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();  
        var response = await cwClient.DescribeAlarmsAsync();  
  
        foreach (var alarm in response.MetricAlarms)  
        {  
            await DescribeAlarmHistoriesAsync(cwClient, alarm.AlarmName);  
        }  
    }  
  
    /// <summary>  
    /// Retrieves the CloudWatch alarm history for the alarm name passed  
    /// to the method.  
    /// </summary>
```

```
    ///<param name="client">An initialized CloudWatch client object.</param>
    ///<param name="alarmName">The CloudWatch alarm for which to retrieve
    /// history information.</param>
    public static async Task DescribeAlarmHistoriesAsync(IAmazonCloudWatch client,
string alarmName)
{
    var request = new DescribeAlarmHistoryRequest
    {
        AlarmName = alarmName,
        EndDateUtc = DateTime.Today,
        HistoryItemType = HistoryItemType.Action,
        MaxRecords = 1,
        StartDateUtc = DateTime.Today.Subtract(TimeSpan.FromDays(30)),
    };

    var response = new DescribeAlarmHistoryResponse();

    do
    {
        response = await client.DescribeAlarmHistoryAsync(request);

        foreach (var item in response.AlarmHistoryItems)
        {
            Console.WriteLine(item.AlarmName);
            Console.WriteLine(item.HistorySummary);
            Console.WriteLine();
        }

        request.NextToken = response.NextToken;
    }
    while (!string.IsNullOrEmpty(response.NextToken));
}
}
```

- For API details, see [DescribeAlarmHistory](#) in *AWS SDK for .NET API Reference*.

## Disable alarm actions

The following code example shows how to disable Amazon CloudWatch alarm actions.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

    ///<summary>
    /// This example shows how to disable the Amazon CloudWatch actions for
    /// one or more CloudWatch alarms. The example was created using the
    /// AWS SDK for .NET version 3.7 and .NET Core 5.0.
    /// </summary>
public class DisableAlarmActions
{
    public static async Task Main()
    {
```

```
IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();
var alarmNames = new List<string>
{
    "ALARM_NAME",
    "ALARM_NAME_2",
};

var success = await DisableAlarmsActionsAsync(cwClient, alarmNames);

if (success)
{
    Console.WriteLine("Alarm action(s) successfully disabled.");
}
else
{
    Console.WriteLine("Alarm action(s) were not disabled.")
}

/// <summary>
/// Disable the actions for the list of CloudWatch alarm names passed
/// in the alarmNames parameter.
/// </summary>
/// <param name="client">An initialized CloudWatch client object.</param>
/// <param name="alarmNames">The list of CloudWatch alarms to disable.</param>
/// <returns>A Boolean value indicating the success of the call.</returns>
public static async Task<bool> DisableAlarmsActionsAsync(
    IAmazonCloudWatch client,
    List<string> alarmNames)
{
    var request = new DisableAlarmActionsRequest
    {
        AlarmNames = alarmNames,
    };

    var response = await client.DisableAlarmActionsAsync(request);

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [DisableAlarmActions](#) in *AWS SDK for .NET API Reference*.

## Enable alarm actions

The following code example shows how to enable Amazon CloudWatch alarm actions.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// This example shows how to enable the Amazon CloudWatch actions for
```

```
/// one or more CloudWatch alarms. The example was created using the
/// AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class EnableAlarmActions
{
    public static async Task Main()
    {
        IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();
        var alarmNames = new List<string>
        {
            "ALARM_NAME",
            "ALARM_NAME_2",
        };

        var success = await EnableAlarmActionsAsync(cwClient, alarmNames);

        if (success)
        {
            Console.WriteLine("Alarm action(s) successfully enabled.");
        }
        else
        {
            Console.WriteLine("Alarm action(s) were not enabled.")
        }
    }

    /// <summary>
    /// Enable the actions for the list of CloudWatch alarm names passed
    /// in the alarmNames parameter.
    /// </summary>
    /// <param name="client">An initialized CloudWatch client object.</param>
    /// <param name="alarmNames">The list of CloudWatch alarms to enable.</param>
    /// <returns>A Boolean value indicating the success of the call.</returns>
    public static async Task<bool> EnableAlarmActionsAsync(IAmazonCloudWatch
client, List<string> alarmNames)
    {
        var request = new EnableAlarmActionsRequest
        {
            AlarmNames = alarmNames,
        };

        var response = await client.EnableAlarmActionsAsync(request);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for .NET API Reference*.

## Get dashboard details

The following code example shows how to get Amazon CloudWatch dashboard details.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
```

```
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// This example shows how to retrieve the details of an Amazon CloudWatch
/// dashboard. The return value from the call to GetDashboard is a json
/// object representing the widgets in the dashboard. The example was
/// created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class GetDashboard
{
    public static async Task Main()
    {
        IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();
        string dashboardName = "CloudWatch-Default";

        var body = await GetDashboardAsync(cwClient, dashboardName);

        Console.WriteLine(body);
    }

    /// <summary>
    /// Get the json that represents the dashboard.
    /// </summary>
    /// <param name="client">An initialized CloudWatch client.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>The string containing the json value describing the
    /// contents and layout of the CloudWatch dashboard.</returns>
    public static async Task<string> GetDashboardAsync(IAmazonCloudWatch client,
string dashboardName)
    {

        var request = new GetDashboardRequest
        {
            DashboardName = dashboardName,
        };

        var response = await client.GetDashboardAsync(request);

        return response.DashboardBody;
    }
}
```

- For API details, see [GetDashboard](#) in *AWS SDK for .NET API Reference*.

## List dashboards

The following code example shows how to list Amazon CloudWatch dashboards.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;
```

```
/// <summary>
/// Shows how to retrieve a list of Amazon CloudWatch dashboards. This
/// example was written using AWSSDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class ListDashboards
{
    public static async Task Main()
    {
        IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();
        var dashboards = await ListDashboardsAsync(cwClient);

        DisplayDashboardList(dashboards);
    }

    /// <summary>
    /// Get the list of available dashboards.
    /// </summary>
    /// <param name="client">The initialized CloudWatch client used to
    /// retrieve a list of defined dashboards.</param>
    /// <returns>A list of DashboardEntry objects.</returns>
    public static async Task<List<DashboardEntry>>
ListDashboardsAsync(IAmazonCloudWatch client)
    {
        var response = await client.ListDashboardsAsync(new
ListDashboardsRequest());
        return response.DashboardEntries;
    }

    /// <summary>
    /// Displays the name of each CloudWatch Dashboard in the list passed
    /// to the method.
    /// </summary>
    /// <param name="dashboards">A list of DashboardEntry objects.</param>
    public static void DisplayDashboardList(List<DashboardEntry> dashboards)
    {
        if (dashboards.Count > 0)
        {
            Console.WriteLine("The following dashboards are defined:");
            foreach (var dashboard in dashboards)
            {
                Console.WriteLine($"Name: {dashboard.DashboardName} Last modified:
{dashboard.LastModified}");
            }
        }
        else
        {
            Console.WriteLine("No dashboards found.");
        }
    }
}
```

- For API details, see [ListDashboards](#) in *AWS SDK for .NET API Reference*.

## List metrics

The following code example shows how to list Amazon CloudWatch metrics.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// This example demonstrates how to list metrics for Amazon CloudWatch.
/// The example was created using the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
/// </summary>
public class ListMetrics
{
    public static async Task Main()
    {
        IAmazonCloudWatch cwClient = new AmazonCloudWatchClient();

        var filter = new DimensionFilter
        {
            Name = "InstanceType",
            Value = "t1.micro",
        };
        string metricName = "CPUUtilization";
        string namespaceName = "AWS/EC2";

        await ListMetricsAsync(cwClient, filter, metricName, namespaceName);
    }

    /// <summary>
    /// Retrieve CloudWatch metrics using the supplied filter, metrics name,
    /// and namespace.
    /// </summary>
    /// <param name="client">An initialized CloudWatch client.</param>
    /// <param name="filter">The filter to apply in retrieving metrics.</param>
    /// <param name="metricName">The metric name for which to retrieve
    /// information.</param>
    /// <param name="nameSpaceName">The name of the namespace from which
    /// to retrieve metric information.</param>
    public static async Task ListMetricsAsync(
        IAmazonCloudWatch client,
        DimensionFilter filter,
        string metricName,
        string nameSpaceName)
    {
        var request = new ListMetricsRequest
        {
            Dimensions = new List<DimensionFilter>() { filter },
            MetricName = metricName,
            Namespace = nameSpaceName,
        };

        var response = new ListMetricsResponse();
        do
        {
            response = await client.ListMetricsAsync(request);

            if (response.Metrics.Count > 0)
            {
                foreach (var metric in response.Metrics)
                {
                    Console.WriteLine(metric.MetricName +
                        " (" + metric.Namespace + ")");

                    foreach (var dimension in metric.Dimensions)

```

```
        {
            Console.WriteLine(" " + dimension.Name + ": "
                + dimension.Value);
        }
    }
else
{
    Console.WriteLine("No metrics found.");
}

request.NextToken = response.NextToken;
}
while (!string.IsNullOrEmpty(response.NextToken));
}
```

- For API details, see [ListMetrics](#) in *AWS SDK for .NET API Reference*.

## CloudWatch Logs examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon CloudWatch Logs.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2280\)](#)

## Actions

### Associate a key with a log group

The following code example shows how to associate an AWS KMS key with an existing CloudWatch Logs log group.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to associate an AWS Key Management Service (AWS KMS) key with
/// an Amazon CloudWatch Logs log group. The example was created using the
/// AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class AssociateKmsKey
```

```
{  
    public static async Task Main()  
    {  
        // This client object will be associated with the same AWS Region  
        // as the default user on this system. If you need to use a  
        // different AWS Region, pass it as a parameter to the client  
        // constructor.  
        var client = new AmazonCloudWatchLogsClient();  
  
        string kmsKeyId = "arn:aws:kms:us-west-2:<account-  
number>:key/7c9ecc2-38cb-4c4f-9db3-766ee8dd3ad4";  
        string groupName = "cloudwatchlogs-example-loggroup";  
  
        var request = new AssociateKmsKeyRequest  
        {  
            KmsKeyId = kmsKeyId,  
            LogGroupName = groupName,  
        };  
  
        var response = await client.AssociateKmsKeyAsync(request);  
  
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)  
        {  
            Console.WriteLine($"Successfully associated KMS key ID: {kmsKeyId} with  
log group: {groupName}.");  
        }  
        else  
        {  
            Console.WriteLine("Could not make the association between: {kmsKeyId}  
and {groupName}.");  
        }  
    }  
}
```

- For API details, see [AssociateKmsKey](#) in *AWS SDK for .NET API Reference*.

## Cancel an export task

The following code example shows how to cancel an existing CloudWatch Logs export task.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Threading.Tasks;  
using Amazon.CloudWatchLogs;  
using Amazon.CloudWatchLogs.Model;  
  
/// <summary>  
/// Shows how to cancel an Amazon CloudWatch Logs export task. The example  
/// uses the AWS SDK for .NET version 3.7 and .NET Core 5.0.  
/// </summary>  
public class CancelExportTask  
{  
    public static async Task Main()  
    {
```

```
// This client object will be associated with the same AWS Region
// as the default user on this system. If you need to use a
// different AWS Region, pass it as a parameter to the client
// constructor.
var client = new AmazonCloudWatchLogsClient();
string taskId = "exampleTaskId";

var request = new CancelExportTaskRequest
{
    TaskId = taskId,
};

var response = await client.CancelExportTaskAsync(request);

if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
{
    Console.WriteLine($"{taskId} successfully canceled.");
}
else
{
    Console.WriteLine($"{taskId} could not be canceled.");
}
}
```

- For API details, see [CancelExportTask](#) in [AWS SDK for .NET API Reference](#).

## Create a log group

The following code example shows how to create a new CloudWatch Logs log group.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Amazon CloudWatch Logs log group. The example
/// was created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class CreateLogGroup
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();

        string logGroupName = "cloudwatchlogs-example-loggroup";

        var request = new CreateLogGroupRequest
        {
```

```
        LogGroupName = logGroupName,
    };

    var response = await client.CreateLogGroupAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"Successfully create log group with ID:
{logGroupName}.");
    }
    else
    {
        Console.WriteLine("Could not create log group.");
    }
}
```

- For API details, see [CreateLogGroup in AWS SDK for .NET API Reference](#).

### Create a new log stream

The following code example shows how to create a new CloudWatch Logs log stream.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

///<summary>
/// Shows how to create an Amazon CloudWatch Logs stream for a CloudWatch
/// log group. The example was created using the AWS SDK for .NET version
/// 3.7 and .NET Core 5.0.
///</summary>
public class CreateLogStream
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string logGroupName = "cloudwatchlogs-example-loggroup";
        string logStreamName = "cloudwatchlogs-example-logstream";

        var request = new CreateLogStreamRequest
        {
            LogGroupName = logGroupName,
            LogStreamName = logStreamName,
        };

        var response = await client.CreateLogStreamAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
        {
            Console.WriteLine($"'{logStreamName}' successfully created for
{logGroupName}.");
        }
        else
        {
            Console.WriteLine("Could not create stream.");
        }
    }
}
```

- For API details, see [CreateLogStream](#) in *AWS SDK for .NET API Reference*.

## Create an export task

The following code example shows how to create a new CloudWatch Logs export task.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to create an Export Task to export the contents of the Amazon
/// CloudWatch Logs to the specified Amazon Simple Storage Service (Amazon S3)
/// bucket. The example was created with the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
/// </summary>
public class CreateExportTask
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();
        string taskId = "export-task-example";
        string logGroupName = "cloudwatchlogs-example-loggroup";
        string destination = "doc-example-bucket";
        var fromTime = 1437584472382;
        var toTime = 1437584472833;

        var request = new CreateExportTaskRequest
        {
            From = fromTime,
            To = toTime,
            TaskName = taskId,
            LogGroupName = logGroupName,
            Destination = destination,
        };

        var response = await client.CreateExportTaskAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
        {
            Console.WriteLine($"The task, {taskName} with ID: " +
                $"'{response.TaskId}' has been created successfully.");
        }
    }
}
```

- For API details, see [CreateExportTask](#) in *AWS SDK for .NET API Reference*.

## Delete a log group

The following code example shows how to delete an existing CloudWatch Logs log group.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

///<summary>
/// Uses the Amazon CloudWatch Logs Service to delete an existing
/// CloudWatch Logs log group. The example was created using the
/// AWS SDK for .NET version 3.7 and .NET Core 5.0.
///</summary>
public class DeleteLogGroup
{
    public static async Task Main()
    {
        var client = new AmazonCloudWatchLogsClient();
        string logGroupName = "cloudwatchlogs-example-loggroup";

        var request = new DeleteLogGroupRequest
        {
            LogGroupName = logGroupName,
        };

        var response = await client.DeleteLogGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully deleted CloudWatch log group,
{logGroupName}.");
        }
    }
}
```

- For API details, see [DeleteLogGroup](#) in *AWS SDK for .NET API Reference*.

## Describe export tasks

The following code example shows how to describe CloudWatch Logs export tasks.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Shows how to retrieve a list of information about Amazon CloudWatch
/// Logs export tasks. The example was created using the AWS SDK for .NET
/// version 3.7 and .NET Core 5.0.
/// </summary>
public class DescribeExportTasks
{
    public static async Task Main()
    {
        // This client object will be associated with the same AWS Region
        // as the default user on this system. If you need to use a
        // different AWS Region, pass it as a parameter to the client
        // constructor.
        var client = new AmazonCloudWatchLogsClient();

        var request = new DescribeExportTasksRequest
        {
            Limit = 5,
        };

        var response = new DescribeExportTasksResponse();

        do
        {
            response = await client.DescribeExportTasksAsync(request);
            response.ExportTasks.ForEach(t =>
            {
                Console.WriteLine($"'{t.TaskName}' with ID: {t.TaskId} has status:
{t.Status}");
            });
        }
        while (response.NextToken is not null);
    }
}
```

- For API details, see [DescribeExportTasks](#) in *AWS SDK for .NET API Reference*.

## Describe log groups

The following code example shows how to describe CloudWatch Logs log groups.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.CloudWatchLogs;
using Amazon.CloudWatchLogs.Model;

/// <summary>
/// Retrieves information about existing Amazon CloudWatch Logs log groups
/// and displays the information on the console. The example was created
/// using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DescribeLogGroups
{
    public static async Task Main()
    {
        // Creates a CloudWatch Logs client using the default
        // user. If you need to work with resources in another
        // AWS Region than the one defined for the default user,
        // pass the AWS Region as a parameter to the client constructor.
        var client = new AmazonCloudWatchLogsClient();

        var request = new DescribeLogGroupsRequest
        {
            Limit = 5,
        };

        var response = await client.DescribeLogGroupsAsync(request);

        if (response.LogGroups.Count > 0)
        {
            do
            {
                response.LogGroups.ForEach(lg =>
                {
                    Console.WriteLine($"'{lg.LogGroupName}' is associated with the
key: {lg.KmsKeyId}.");
                    Console.WriteLine($"Created on: {lg.CreationTime.Date.Date}");
                    Console.WriteLine($"Date for this group will be stored for:
{lg.RetentionInDays} days.\n");
                });
            }
            while (response.NextToken is not null);
        }
    }
}
```

- For API details, see [DescribeLogGroups](#) in *AWS SDK for .NET API Reference*.

## Amazon Cognito Identity Provider examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon Cognito Identity Provider.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2288\)](#)

- [Scenarios \(p. 2295\)](#)

## Actions

### Confirm a user

The following code example shows how to confirm an Amazon Cognito user.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Confirms that a user has been signed up successfully.
///</summary>
///<param name="identityProviderClient">An initialized Identity
/// Provider client object.</param>
///<param name="clientId">The client Id of the application associated
/// with the user pool.</param>
///<param name="code">The code sent by the authentication provider
/// to confirm a user's membership in the pool.</param>
///<param name="userName">The user to confirm.</param>
///<returns>A Boolean value indicating the success of the confirmation
/// operation.</returns>
public static async Task<bool> ConfirmSignUp(
    AmazonCognitoIdentityProviderClient identityProviderClient,
    string clientId,
    string code,
    string userName)
{
    var signUpRequest = new ConfirmSignUpRequest
    {
        ClientId = clientId,
        ConfirmationCode = code,
        Username = userName,
    };

    var response = await
identityProviderClient.ConfirmSignUpAsync(signUpRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{userName} was confirmed");
        return true;
    }
    else
    {
        return false;
    }
}
```

- For API details, see [ConfirmSignUp in AWS SDK for .NET API Reference](#).

### Get a token to associate an MFA application with a user

The following code example shows how to get a token to associate an MFA application with an Amazon Cognito user.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Gets the secret token that will enable multi-factor
/// authentication (MFA) for the user.
/// </summary>
/// <param name="identityProviderClient">An initialized Identity
/// Provider client object.</param>
/// <param name="session">The currently active session.</param>
/// <returns>Returns a string representing the currently active
/// session.</returns>
public static async Task<string> GetSecretForAppMFA(
    AmazonCognitoIdentityProviderClient identityProviderClient,
    string session)
{
    var softwareTokenRequest = new AssociateSoftwareTokenRequest
    {
        Session = session,
    };

    var tokenResponse = await
identityProviderClient.AssociateSoftwareTokenAsync(softwareTokenRequest);
    var secretCode = tokenResponse.SecretCode;

    Console.WriteLine($"Enter the following token into Google Authenticator:
{secretCode}");

    return tokenResponse.Session;
}
```

- For API details, see [AssociateSoftwareToken](#) in *AWS SDK for .NET API Reference*.

## Get information about a user

The following code example shows how to get information about an Amazon Cognito user.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Checks the status of a user for a particular Amazon Cognito user
/// pool.
/// </summary>
/// <param name="identityProviderClient">An initialized Identity
/// Provider client object.</param>
/// <param name="userName">The user name for which we want to check
/// the status.</param>
/// <param name="poolId">The user pool for which we want to check the
/// user's status.</param>
```

```
    /// <returns>The UserStatusType object indicating the user's status.</returns>
    public static async Task<UserStatusType>
GetAdminUser(AmazonCognitoIdentityProviderClient identityProviderClient, string
userName, string poolId)
{
    var userRequest = new Admin GetUserRequest
    {
        Username = userName,
        UserPoolId = poolId,
    };

    var response = await identityProviderClient.Admin GetUserAsync(userRequest);

    Console.WriteLine($"User status {response.UserStatus}");
    return response.UserStatus;
}
```

- For API details, see [Admin GetUser](#) in *AWS SDK for .NET API Reference*.

## Resend a confirmation code

The following code example shows how to resend an Amazon Cognito confirmation code.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    /// <summary>
    /// Causes the confirmation code for user registration to be sent
    /// again.
    /// </summary>
    /// <param name="identityProviderClient">An initialized Identity
    /// Provider client object.</param>
    /// <param name="clientId">The client Id of the application associated
    /// with the user pool.</param>
    /// <param name="userName">The user name to be confirmed.</param>
    /// <returns>A System Threading Task.</returns>
    public static async Task
ResendConfirmationCode(AmazonCognitoIdentityProviderClient identityProviderClient,
string clientId, string userName)
{
    var codeRequest = new ResendConfirmationCodeRequest
    {
        ClientId = clientId,
        Username = userName,
    };

    var response = await
identityProviderClient.ResendConfirmationCodeAsync(codeRequest);

    Console.WriteLine($"Method of delivery is
{response.CodeDeliveryDetails.DeliveryMedium}");
}
```

- For API details, see [ResendConfirmationCode](#) in *AWS SDK for .NET API Reference*.

## Respond to SRP authentication challenges

The following code example shows how to respond to Amazon Cognito SRP authentication challenges.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Responds to an authentication challenge for an Amazon Cognito user.
/// </summary>
/// <param name="identityProviderClient">The Amazon Cognito client object.</param>
/// <param name="userName">The user name of the user to authenticate.</param>
/// <param name="clientId">The client Id of the application associated with the user pool.</param>
/// <param name="mfaCode">The MFA code supplied by the user.</param>
/// <param name="session">The session for which the user will be authenticated.</param>
/// <returns>A Boolean value that indicates the success of the authentication.</returns>
public static async Task<bool> AdminRespondToAuthChallenge(
    AmazonCognitoIdentityProviderClient identityProviderClient,
    string userName,
    string clientId,
    string mfaCode,
    string session)
{
    Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

    var challengeResponses = new Dictionary<string, string>
    {
        { "USERNAME", userName },
        { "SOFTWARE_TOKEN_MFA_CODE", mfaCode },
    };

    var respondToAuthChallengeRequest = new RespondToAuthChallengeRequest
    {
        ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
        ClientId = clientId,
        ChallengeResponses = challengeResponses,
        Session = session,
    };

    var response = await
identityProviderClient.RespondToAuthChallengeAsync(respondToAuthChallengeRequest);
    Console.WriteLine($"response.getAuthenticationResult() {response.AuthenticationResult}");

    return response.AuthenticationResult is not null;
}
```

- For API details, see [RespondToAuthChallenge](#) in *AWS SDK for .NET API Reference*.

## Respond to an authentication challenge

The following code example shows how to respond to an Amazon Cognito authentication challenge.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Responds to an authentication challenge for an Amazon Cognito user.
/// </summary>
/// <param name="identityProviderClient">The Amazon Cognito client object.</param>
/// <param name="userName">The user name of the user to authenticate.</param>
/// <param name="clientId">The client Id of the application associated with the user pool.</param>
/// <param name="mfaCode">The MFA code supplied by the user.</param>
/// <param name="session">The session for which the user will be authenticated.</param>
/// <returns>A Boolean value that indicates the success of the authentication.</returns>
public static async Task<bool> AdminRespondToAuthChallenge(
    AmazonCognitoIdentityProviderClient identityProviderClient,
    string userName,
    string clientId,
    string mfaCode,
    string session)
{
    Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

    var challengeResponses = new Dictionary<string, string>
    {
        { "USERNAME", userName },
        { "SOFTWARE_TOKEN_MFA_CODE", mfaCode },
    };

    var respondToAuthChallengeRequest = new RespondToAuthChallengeRequest
    {
        ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
        ClientId = clientId,
        ChallengeResponses = challengeResponses,
        Session = session,
    };

    var response = await
identityProviderClient.RespondToAuthChallengeAsync(respondToAuthChallengeRequest);
    Console.WriteLine($"response.getAuthenticationResult() {response.AuthenticationResult}");

    return response.AuthenticationResult is not null;
}
```

- For API details, see [AdminRespondToAuthChallenge](#) in *AWS SDK for .NET API Reference*.

## Sign up a user

The following code example shows how to sign up a user with Amazon Cognito.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Add a new user to an Amazon Cognito user pool.
/// </summary>
/// <param name="identityProviderClient">An initialized Identity
/// Provider client object.</param>
/// <param name="clientId">The client Id of the application associated
/// with the user pool.</param>
/// <param name="userName">The user name of the user to sign up.</param>
/// <param name="password">The password for the user.</param>
/// <param name="email">The user's email address.</param>
/// <returns>A System Threading Task<string> SignUp(
    public static async Task<string> SignUp(
        AmazonCognitoIdentityProviderClient identityProviderClient,
        string clientId,
        string userName,
        string password,
        string email)
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>
        {
            userAttrs,
        };

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password,
        };

        var response = await identityProviderClient.SignUpAsync(signUpRequest);
        Console.WriteLine("User has been signed up.");
        return response.UserSub;
    }
}
```

- For API details, see [SignUp](#) in *AWS SDK for .NET API Reference*.

## Start authentication with a tracked device

The following code example shows how to start authentication with a device tracked by Amazon Cognito.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    /// <summary>
    /// Initiates the authorization process.
    /// </summary>
    /// <param name="identityProviderClient">An initialized Identity
    /// Provider client object.</param>
    /// <param name="clientId">The client Id of the application associated
    /// with the user pool.</param>
    /// <param name="userName">The user name to be authorized.</param>
    /// <param name="password">The password of the user.</param>
    /// <returns>The response from the client from the InitiateAuthAsync
    /// call.</returns>
    public static async Task<InitiateAuthResponse>
InitiateAuth(AmazonCognitoIdentityProviderClient identityProviderClient, string
clientId, string userName, string password)
{
    var authParameters = new Dictionary<string, string>
    {
        { "USERNAME", userName },
        { "PASSWORD", password },
    };

    var authRequest = new InitiateAuthRequest
    {
        ClientId = clientId,
        AuthParameters = authParameters,
        AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
    };

    var response = await identityProviderClient.InitiateAuthAsync(authRequest);
    Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

    return response;
}
```

- For API details, see [InitiateAuth](#) in *AWS SDK for .NET API Reference*.

## Verify an MFA application with a user

The following code example shows how to verify an MFA application with an Amazon Cognito user.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    /// <summary>
    /// Verifies that the user has supplied the correct one-time password
    /// and registers for multi-factor authentication (MFA).
    /// </summary>
    /// <param name="identityProviderClient">The Amazon Cognito client object.</
param>
    /// <param name="session">The session for which the user will be
    /// authenticated.</param>
    /// <param name="code">The code provided by the user.</param>
    /// <returns>A Boolean value that indicates the success of the
    /// authentication.</returns>
    public static async Task<bool> VerifyTOTP(
```

```
        AmazonCognitoIdentityProviderClient identityProviderClient,
        string session,
        string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var response = await
identityProviderClient.VerifySoftwareTokenAsync(tokenRequest);

        Console.WriteLine($"The status of the token is {response.Status}");

        return response.Status == VerifySoftwareTokenResponseType.SUCCESS;
    }
}
```

- For API details, see [VerifySoftwareToken](#) in *AWS SDK for .NET API Reference*.

## Scenarios

### Sign up a user with a user pool that requires MFA

The following code example shows how to:

- Sign up a user with a user name, password, and email address.
- Confirm the user from a code sent in email.
- Set up multi-factor authentication by associating an MFA application with the user.
- Sign in by using a password and an MFA code.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
global using Amazon;
global using Amazon.CognitoIdentityProvider;
global using Amazon.CognitoIdentityProvider.Model;
global using Cognito_MVP;

// Before running this AWS SDK for .NET (v3) code example, set up your development
// environment, including your credentials.
// For more information, see the following documentation:
// https://docs.aws.amazon.com/sdk-for-net/v3/developer-guide/net-dg-setup.html
// TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS CDK)
// script provided in this GitHub repo at:
// resources/cdk/cognito_scenario_user_pool_with_mfa.
// This code example performs the following operations:
// 1. Invokes the signUp method to sign up a user.
// 2. Invokes the adminGetUser method to get the user's confirmation status.
// 3. Invokes the ResendConfirmationCode method if the user requested another code.
// 4. Invokes the confirmSignUp method.
// 5. Invokes the initiateAuth to sign in. This results in being prompted to set up
// TOTP (time-based one-time password). (The response is "ChallengeName": "MFA_SETUP").
```

```
// 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.  
// This can be used with Google Authenticator.  
// 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for MFA.  
// 8. Invokes the AdminInitiateAuth to sign in again. This results in being prompted to  
// submit a TOTP (Response: "ChallengeName": "SOFTWARE_TOKEN_MFA").  
// 9. Invokes the AdminRespondToAuthChallenge to get back a token.  
  
// Set the following variables:  
// clientId - Fill in the app client Id value from the AWS CDK script.  
string clientId = "";  
  
// poolId - Fill in the pool Id that you can get from the AWS CDK script.  
string poolId = "";  
var userName = string.Empty;  
var password = string.Empty;  
var email = string.Empty;  
  
string sepBar = new string('-', 80);  
var identityProviderClient = new AmazonCognitoIdentityProviderClient();  
  
do  
{  
    Console.Write("Enter your user name: ");  
    userName = Console.ReadLine();  
}  
while (userName == string.Empty);  
  
Console.WriteLine($"User name: {userName}");  
  
do  
{  
    Console.Write("Enter your password: ");  
    password = Console.ReadLine();  
}  
while (password == string.Empty);  
Console.WriteLine($"Signing up {userName}");  
  
do  
{  
    Console.Write("Enter your email: ");  
    email = Console.ReadLine();  
} while (email == string.Empty);  
  
await CognitoMethods.SignUp(identityProviderClient, clientId, userName, password,  
    email);  
  
Console.WriteLine(sepBar);  
Console.WriteLine($"Getting {userName} status from the user pool");  
await CognitoMethods.GetAdminUser(identityProviderClient, userName, poolId);  
  
Console.WriteLine(sepBar);  
Console.WriteLine($"Conformation code sent to {userName}. Would you like to send a new  
code? (Yes/No)");  
var ans = Console.ReadLine();  
  
if (ans.ToUpper() == "YES")  
{  
    await CognitoMethods.ResendConfirmationCode(identityProviderClient, clientId,  
        userName);  
    Console.WriteLine("Sending a new confirmation code");  
}  
  
Console.WriteLine(sepBar);  
Console.WriteLine("*** Enter confirmation code that was emailed");  
string code = Console.ReadLine();
```

```
await CognitoMethods.ConfirmSignUp(identityProviderClient, clientId, code, userName);

Console.WriteLine($"Rechecking the status of {userName} in the user pool");
await CognitoMethods.GetAdminUser(identityProviderClient, userName, poolId);

var authResponse = await CognitoMethods.InitiateAuth(identityProviderClient, clientId,
    userName, password);
var mySession = authResponse.Session;

var newSession = await CognitoMethods.GetSecretForAppMFA(identityProviderClient,
    mySession);

Console.WriteLine("Enter the 6-digit code displayed in Google Authenticator");
string myCode = Console.ReadLine();

// Verify the TOTP and register for MFA.
await CognitoMethods.VerifyTOTP(identityProviderClient, newSession, myCode);
Console.WriteLine("Enter the new 6-digit code displayed in Google Authenticator");
string mfaCode = Console.ReadLine();

Console.WriteLine(sepBar);
var authResponse1 = await CognitoMethods.InitiateAuth(identityProviderClient, clientId,
    userName, password);
var session2 = authResponse1.Session;
await CognitoMethods.AdminRespondToAuthChallenge(identityProviderClient, userName,
    clientId, mfaCode, session2);

Console.WriteLine("The Cognito MVP application has completed.");
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [Admin GetUser](#)
  - [Admin Initiate Auth](#)
  - [Admin Respond To Auth Challenge](#)
  - [Associate Software Token](#)
  - [Confirm Device](#)
  - [Confirm Sign Up](#)
  - [Initiate Auth](#)
  - [List Users](#)
  - [Resend Confirmation Code](#)
  - [Respond To Auth Challenge](#)
  - [Sign Up](#)
  - [Verify Software Token](#)

## Amazon Comprehend examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon Comprehend.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2298\)](#)

## Actions

### Detect entities in a document

The following code example shows how to detect entities in a document with Amazon Comprehend.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

///<summary>
/// This example shows how to use the AmazonComprehend service detect any
/// entities in submitted text. This example was created using the AWS SDK
/// for .NET 3.7 and .NET Core 5.0.
///</summary>
public static class DetectEntities
{
    ///<summary>
    /// The main method calls the DetectEntitiesAsync method to find any
    /// entities in the sample code.
    ///</summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle";

        var comprehendClient = new AmazonComprehendClient();

        Console.WriteLine("Calling DetectEntities\n");
        var detectEntitiesRequest = new DetectEntitiesRequest()
        {
            Text = text,
            LanguageCode = "en",
        };
        var detectEntitiesResponse = await
comprehendClient.DetectEntitiesAsync(detectEntitiesRequest);

        foreach (var e in detectEntitiesResponse.Entities)
        {
            Console.WriteLine($"Text: {e.Text}, Type: {e.Type}, Score: {e.Score},
BeginOffset: {e.BeginOffset}, EndOffset: {e.EndOffset}");
        }

        Console.WriteLine("Done");
    }
}
```

- For API details, see [DetectEntities](#) in *AWS SDK for .NET API Reference*.

### Detect key phrases in a document

The following code example shows how to detect key phrases in a document with Amazon Comprehend.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to use the Amazon Comprehend service to
/// search text for key phrases. It was created using the AWS SDK for
/// .NET version 3.7 and .NET Core 5.0.
/// </summary>
public static class DetectKeyPhrase
{
    /// <summary>
    /// This method calls the Amazon Comprehend method DetectKeyPhrasesAsync
    /// to detect any key phrases in the sample text.
    /// </summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle";

        var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

        // Call DetectKeyPhrases API
        Console.WriteLine("Calling DetectKeyPhrases");
        var detectKeyPhrasesRequest = new DetectKeyPhrasesRequest()
        {
            Text = text,
            LanguageCode = "en",
        };
        var detectKeyPhrasesResponse = await
comprehendClient.DetectKeyPhrasesAsync(detectKeyPhrasesRequest);
        foreach (var kp in detectKeyPhrasesResponse.KeyPhrases)
        {
            Console.WriteLine($"Text: {kp.Text}, Score: {kp.Score}, BeginOffset:
{kp.BeginOffset}, EndOffset: {kp.EndOffset}");
        }

        Console.WriteLine("Done");
    }
}
```

- For API details, see [DetectKeyPhrases](#) in *AWS SDK for .NET API Reference*.

## Detect personally identifiable information in a document

The following code example shows how to detect personally identifiable information (PII) in a document with Amazon Comprehend.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

///<summary>
/// This example shows how to use the Amazon Comprehend service to find
/// personally identifiable information (PII) within text submitted to the
/// DetectPiiEntitiesAsync method. The example was created using the AWS
/// SDK for .NET version 3.7 and .NET Core 5.0.
///</summary>
public class DetectingPII
{
    ///<summary>
    /// This method calls the DetectPiiEntitiesAsync method to locate any
    /// personally identifiable information within the supplied text.
    ///</summary>
    public static async Task Main()
    {
        var comprehendClient = new AmazonComprehendClient();
        var text = @"Hello Paul Santos. The latest statement for your
                    credit card account 1111-0000-1111-0000 was
                    mailed to 123 Any Street, Seattle, WA 98109.";

        var request = new DetectPiiEntitiesRequest
        {
            Text = text,
            LanguageCode = "EN",
        };

        var response = await comprehendClient.DetectPiiEntitiesAsync(request);

        if (response.Entities.Count > 0)
        {
            foreach (var entity in response.Entities)
            {
                var entityValue = text.Substring(entity.BeginOffset,
entity.EndOffset - entity.BeginOffset);
                Console.WriteLine($"{entity.Type}: {entityValue}");
            }
        }
    }
}
```

- For API details, see [DetectPiiEntities in AWS SDK for .NET API Reference](#).

## Detect syntactical elements of a document

The following code example shows how to detect syntactical elements of a document with Amazon Comprehend.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
```

```
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to use Amazon Comprehend to detect syntax
/// elements by calling the DetectSyntaxAsync method. This example was
/// created using the AWS SDK for .NET 3.7 and .NET Core 5.0.
/// </summary>
public class DetectingSyntax
{
    /// <summary>
    /// This method calls DetectSynaxAsync to identify the syntax elements
    /// in the sample text.
    /// </summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle";

        var comprehendClient = new AmazonComprehendClient();

        // Call DetectSyntax API
        Console.WriteLine("Calling DetectSyntaxAsync\n");
        var detectSyntaxRequest = new DetectSyntaxRequest()
        {
            Text = text,
            LanguageCode = "en",
        };
        DetectSyntaxResponse detectSyntaxResponse = await
comprehendClient.DetectSyntaxAsync(detectSyntaxRequest);
        foreach (SyntaxToken s in detectSyntaxResponse.SyntaxTokens)
        {
            Console.WriteLine($"Text: {s.Text}, PartOfSpeech: {s.PartOfSpeech.Tag},
BeginOffset: {s.BeginOffset}, EndOffset: {s.EndOffset}");
        }

        Console.WriteLine("Done");
    }
}
```

- For API details, see [DetectSyntax](#) in *AWS SDK for .NET API Reference*.

## Detect the dominant language in a document

The following code example shows how to detect the dominant language in a document with Amazon Comprehend.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example calls the Amazon Comprehend service to determine the
```

```
/// dominant language. The example was created using the AWS SDK for .NET
/// 3.7 and .NET Core 5.0.
/// </summary>
public static class DetectDominantLanguage
{
    /// <summary>
    /// Calls Amazon Comprehend to determine the dominant language used in
    /// the sample text.
    /// </summary>
    public static async Task Main()
    {
        string text = "It is raining today in Seattle.";

        var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

        Console.WriteLine("Calling DetectDominantLanguage\n");
        var detectDominantLanguageRequest = new DetectDominantLanguageRequest()
        {
            Text = text,
        };

        var detectDominantLanguageResponse = await
comprehendClient.DetectDominantLanguageAsync(detectDominantLanguageRequest);
        foreach (var dl in detectDominantLanguageResponse.Languages)
        {
            Console.WriteLine($"Language Code: {dl.LanguageCode}, Score:
{dl.Score}");
        }

        Console.WriteLine("Done");
    }
}
```

- For API details, see [DetectDominantLanguage in AWS SDK for .NET API Reference](#).

## Detect the sentiment of a document

The following code example shows how to detect the sentiment of a document with Amazon Comprehend.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example shows how to detect the overall sentiment of the supplied
/// text using the Amazon Comprehend service. The example was writing using
/// the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public static class DetectSentiment
{
    /// <summary>
```

```
/// This method calls the DetectSentimentAsync method to analyze the
/// supplied text and determine the overall sentiment.
/// </summary>
public static async Task Main()
{
    string text = "It is raining today in Seattle";

    var comprehendClient = new
AmazonComprehendClient(Amazon.RegionEndpoint.USWest2);

    // Call DetectKeyPhrases API
    Console.WriteLine("Calling DetectSentiment");
    var detectSentimentRequest = new DetectSentimentRequest()
    {
        Text = text,
        LanguageCode = "en",
    };
    var detectSentimentResponse = await
comprehendClient.DetectSentimentAsync(detectSentimentRequest);
    Console.WriteLine($"Sentiment: {detectSentimentResponse.Sentiment}");
    Console.WriteLine("Done");
}
}
```

- For API details, see [DetectSentiment](#) in *AWS SDK for .NET API Reference*.

## Start a topic modeling job

The following code example shows how to start an Amazon Comprehend topic modeling job.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Comprehend;
using Amazon.Comprehend.Model;

/// <summary>
/// This example scans the documents in an Amazon Simple Storage Service
/// (Amazon S3) bucket and analyzes it for topics. The results are stored
/// in another bucket and then the resulting job properties are displayed
/// on the screen. This example was created using the AWS SDK for .NET
/// version 3.7 and .NET Core version 5.0.
/// </summary>
public static class TopicModeling
{
    /// <summary>
    /// This method calls a topic detection job by calling the Amazon
    /// Comprehend StartTopicsDetectionJobRequest.
    /// </summary>
    public static async Task Main()
    {
        var comprehendClient = new AmazonComprehendClient();

        string inputS3Uri = "s3://input bucket/input path";
        InputFormat inputDocFormat = InputFormat.ONE_DOC_PER_FILE;
```

```
string outputS3Uri = "s3://output bucket/output path";
string dataAccessRoleArn = "arn:aws:iam::account ID:role/data access role";
int numberOfWorkTopics = 10;

var startTopicsDetectionJobRequest = new StartTopicsDetectionJobRequest()
{
    InputDataConfig = new InputDataConfig()
    {
        S3Uri = inputS3Uri,
        InputFormat = inputDocFormat,
    },
    OutputDataConfig = new OutputDataConfig()
    {
        S3Uri = outputS3Uri,
    },
    DataAccessRoleArn = dataAccessRoleArn,
    NumberOfTopics = numberOfWorkTopics,
};

var startTopicsDetectionJobResponse = await
comprehendClient.StartTopicsDetectionJobAsync(startTopicsDetectionJobRequest);

var jobId = startTopicsDetectionJobResponse.JobId;
Console.WriteLine("JobId: " + jobId);

var describeTopicsDetectionJobRequest = new
DescribeTopicsDetectionJobRequest()
{
    JobId = jobId,
};

var describeTopicsDetectionJobResponse = await
comprehendClient.DescribeTopicsDetectionJobAsync(describeTopicsDetectionJobRequest);

PrintJobProperties(describeTopicsDetectionJobResponse.TopicsDetectionJobProperties);

var listTopicsDetectionJobsResponse = await
comprehendClient.ListTopicsDetectionJobsAsync(new ListTopicsDetectionJobsRequest());
foreach (var props in
listTopicsDetectionJobsResponse.TopicsDetectionJobPropertiesList)
{
    PrintJobProperties(props);
}

/// <summary>
/// This method is a helper method that displays the job properties
/// from the call to StartTopicsDetectionJobRequest.
/// </summary>
/// <param name="props">A list of properties from the call to
/// StartTopicsDetectionJobRequest.</param>
private static void PrintJobProperties(TopicsDetectionJobProperties props)
{
    Console.WriteLine($"JobId: {props.JobId}, JobName: {props.JobName},
JobStatus: {props.JobStatus}");
    Console.WriteLine($"NumberOfTopics: {props.NumberOfTopics}\nInputS3Uri:
{props.InputDataConfig.S3Uri}");
    Console.WriteLine($"InputFormat: {props.InputDataConfig.InputFormat},
OutputS3Uri: {props.OutputDataConfig.S3Uri}");
}
```

- For API details, see [StartTopicsDetectionJob](#) in *AWS SDK for .NET API Reference*.

## DynamoDB examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon DynamoDB.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2305\)](#)
- [Scenarios \(p. 2320\)](#)

## Actions

### Create a table

The following code example shows how to create a DynamoDB table.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Creates a new Amazon DynamoDB table and then waits for the new
/// table to become active.
///</summary>
///<param name="client">An initialized Amazon DynamoDB client object.</param>
///<param name="tableName">The name of the table to create.</param>
///<returns>A Boolean value indicating the success of the operation.</returns>
public static async Task<bool> CreateMovieTableAsync(AmazonDynamoDBClient
client, string tableName)
{
    var response = await client.CreateTableAsync(new CreateTableRequest
    {
        TableName = tableName,
        AttributeDefinitions = new List<AttributeDefinition>()
        {
            new AttributeDefinition
            {
                AttributeName = "title",
                AttributeType = "S",
            },
            new AttributeDefinition
            {
                AttributeName = "year",
                AttributeType = "N",
            },
        },
        KeySchema = new List<KeySchemaElement>()
        {
            new KeySchemaElement
            {
                AttributeName = "year",
                KeyType = "HASH",
            }
        }
    });
    return response.TableStatus == CreateTableResponse.TableStatus.SUCCEEDED;
}
```

```
        },
        new KeySchemaElement
        {
            AttributeName = "title",
            KeyType = "RANGE",
        },
    },
    ProvisionedThroughput = new ProvisionedThroughput
    {
        ReadCapacityUnits = 5,
        WriteCapacityUnits = 5,
    },
);

// Wait until the table is ACTIVE and then report success.
Console.WriteLine("Waiting for table to become active...");

var request = new DescribeTableRequest
{
    TableName = response.TableDescription.TableName,
};

TableStatus status;

int sleepDuration = 2000;

do
{
    System.Threading.Thread.Sleep(sleepDuration);

    var describeTableResponse = await client.DescribeTableAsync(request);
    status = describeTableResponse.Table.TableStatus;

    Console.WriteLine(".");
}
while (status != "ACTIVE");

return status == TableStatus.ACTIVE;
}
```

- For API details, see [CreateTable](#) in *AWS SDK for .NET API Reference*.

## Delete a table

The following code example shows how to delete a DynamoDB table.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static async Task<bool> DeleteTableAsync(AmazonDynamoDBClient client,
string tableName)
{
    var request = new DeleteTableRequest
    {
        TableName = tableName,
    };
}
```

```
        var response = await client.DeleteTableAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Table {response.TableDescription.TableName}
successfully deleted.");
            return true;
        }
        else
        {
            Console.WriteLine("Could not delete table.");
            return false;
        }
    }
```

- For API details, see [DeleteTable](#) in *AWS SDK for .NET API Reference*.

## Delete an item from a table

The following code example shows how to delete an item from a DynamoDB table.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Deletes a single item from a DynamoDB table.
/// </summary>
/// <param name="client">The initialized DynamoDB client object.</param>
/// <param name="tableName">The name of the table from which the item
/// will be deleted.</param>
/// <param name="movieToDelete">A movie object containing the title and
/// year of the movie to delete.</param>
/// <returns>A Boolean value indicating the success or failure of the
/// delete operation.</returns>
public static async Task<bool> DeleteItemAsync(
    AmazonDynamoDBClient client,
    string tableName,
    Movie movieToDelete)
{
    var key = new Dictionary<string, AttributeValue>
    {
        ["title"] = new AttributeValue { S = movieToDelete.Title },
        ["year"] = new AttributeValue { N = movieToDelete.Year.ToString() },
    };

    var request = new DeleteItemRequest
    {
        TableName = tableName,
        Key = key,
    };

    var response = await client.DeleteItemAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [DeleteItem](#) in *AWS SDK for .NET API Reference*.

## Get a batch of items

The following code example shows how to get a batch of DynamoDB items.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Gets information about an existing movie from the table.
/// </summary>
/// <param name="client">An initialized Amazon DynamoDB client object.</param>
/// <param name="newMovie">A Movie object containing information about
/// the movie to retrieve.</param>
/// <param name="tableName">The name of the table containing the movie.</param>
/// <returns>A Dictionary object containing information about the item
/// retrieved.</returns>
public static async Task<Dictionary<string, AttributeValue>>
GetItemAsync(AmazonDynamoDBClient client, Movie newMovie, string tableName)
{
    var key = new Dictionary<string, AttributeValue>
    {
        ["title"] = new AttributeValue { S = newMovie.Title },
        ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
    };

    var request = new GetItemRequest
    {
        Key = key,
        TableName = tableName,
    };

    var response = await client.GetItemAsync(request);
    return response.Item;
}
```

- For API details, see [BatchGetItem](#) in *AWS SDK for .NET API Reference*.

## Put an item in a table

The following code example shows how to put an item in a DynamoDB table.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Adds a new item to the table.
/// </summary>
```

```
    ///<param name="client">An initialized Amazon DynamoDB client object.</param>
    ///<param name="newMovie">A Movie object containing information for
    ///the movie to add to the table.</param>
    ///<param name="tableName">The name of the table where the item will be
    added.</param>
    ///<returns>A Boolean value that indicates the results of adding the item.</returns>
    public static async Task<bool> PutItemAsync(AmazonDynamoDBClient client, Movie
    newMovie, string tableName)
    {
        var item = new Dictionary<string, AttributeValue>
        {
            ["title"] = new AttributeValue { S = newMovie.Title },
            ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
        };

        var request = new PutItemRequest
        {
            TableName = tableName,
            Item = item,
        };

        var response = await client.PutItemAsync(request);
        return response.HttpStatusCode == System.Net HttpStatusCode.OK;
    }
```

- For API details, see [PutItem](#) in *AWS SDK for .NET API Reference*.

## Query a table

The following code example shows how to query a DynamoDB table.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    ///<summary>
    /// Queries the table for movies released in a particular year and
    /// then displays the information for the movies returned.
    ///</summary>
    ///<param name="client">The initialized DynamoDB client object.</param>
    ///<param name="tableName">The name of the table to query.</param>
    ///<param name="year">The release year for which we want to
    ///view movies.</param>
    ///<returns>The number of movies that match the query.</returns>
    public static async Task<int> QueryMoviesAsync(AmazonDynamoDBClient client,
    string tableName, int year)
    {
        var movieTable = Table.LoadTable(client, tableName);
        var filter = new QueryFilter("year", QueryOperator.Equal, year);

        Console.WriteLine("\nFind movies released in: {year}:");

        var config = new QueryOperationConfig()
        {
            Limit = 10, // 10 items per page.
            Select = SelectValues.SpecificAttributes,
```

```
        AttributesToGet = new List<string>
    {
        "title",
        "year",
    },
    ConsistentRead = true,
    Filter = filter,
};

// Value used to track how many movies match the
// supplied criteria.
var moviesFound = 0;

Search search = movieTable.Query(config);
do
{
    var movieList = await search.GetNextSetAsync();
    moviesFound += movieList.Count;

    foreach (var movie in movieList)
    {
        DisplayDocument(movie);
    }
}
while (!search.IsDone);

return moviesFound;
}
```

- For API details, see [Query in AWS SDK for .NET API Reference](#).

## Run a PartiQL statement

The following code example shows how to run a PartiQL statement on a DynamoDB table.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use an INSERT statement to add an item.

```
/// <summary>
/// Inserts a single movie into the movies table.
/// </summary>
/// <param name="tableName">The name of the table.</param>
/// <param name="movieTitle">The title of the movie to insert.</param>
/// <param name="year">The year that the movie was released.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the INSERT operation.</returns>
public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
{
    string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?, 'year': ?}}";
    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
```

```
        Statement = insertBatch,
        Parameters = new List<AttributeValue>
        {
            new AttributeValue { S = movieTitle },
            new AttributeValue { N = year.ToString() },
        },
    });

    return response.StatusCode == System.Net.HttpStatusCode.OK;
}
```

Use a SELECT statement to get an item.

```
/// <summary>
/// Uses a PartiQL SELECT statement to retrieve a single movie from the
/// movie database.
/// </summary>
/// <param name="tableName">The name of the movie table.</param>
/// <param name="movieTitle">The title of the movie to retrieve.</param>
/// <returns>A list of movie data. If no movie matches the supplied
/// title, the list is empty.</returns>
public static async Task<List<Dictionary<string, AttributeValue>>>
GetSingleMovie(string tableName, string movieTitle)
{
    string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
    var parameters = new List<AttributeValue>
    {
        new AttributeValue { S = movieTitle },
    };

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = selectSingle,
    Parameters = parameters,
});

    return response.Items;
}
```

Use a SELECT statement to get a list of items.

```
/// <summary>
/// Retrieve multiple movies by year using a SELECT statement.
/// </summary>
/// <param name="tableName">The name of the movie table.</param>
/// <param name="year">The year the movies were released.</param>
/// <returns></returns>
public static async Task<List<Dictionary<string, AttributeValue>>>
GetMovies(string tableName, int year)
{
    string selectSingle = $"SELECT * FROM {tableName} WHERE year = ?";
    var parameters = new List<AttributeValue>
    {
        new AttributeValue { N = year.ToString() },
    };
}
```

```
        var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = selectSingle,
    Parameters = parameters,
});

return response.Items;
}
```

Use an UPDATE statement to update an item.

```
/// <summary>
/// Updates a single movie in the table, adding information for the
/// producer.
/// </summary>
/// <param name="tableName">the name of the table.</param>
/// <param name="producer">The name of the producer.</param>
/// <param name="movieTitle">The movie title.</param>
/// <param name="year">The year the movie was released.</param>
/// <returns>A Boolean value that indicates the success of the
/// UPDATE operation.</returns>
public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
{
    string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title = ?
AND year = ?";

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = insertSingle,
    Parameters = new List<AttributeValue>
    {
        new AttributeValue { S = producer },
        new AttributeValue { S = movieTitle },
        new AttributeValue { N = year.ToString() },
    },
});
    return response.StatusCode == System.Net.HttpStatusCode.OK;
}
```

Use a DELETE statement to delete a single movie.

```
/// <summary>
/// Deletes a single movie from the table.
/// </summary>
/// <param name="tableName">The name of the table.</param>
/// <param name="movieTitle">The title of the movie to delete.</param>
/// <param name="year">The year that the movie was released.</param>
/// <returns>A Boolean value that indicates the success of the
/// DELETE operation.</returns>
public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
{
    var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year = ?";
```

```
        var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = deleteSingle,
    Parameters = new List<AttributeValue>
    {
        new AttributeValue { S = movieTitle },
        new AttributeValue { N = year.ToString() },
    },
});
return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for .NET API Reference*.

## Run batches of PartiQL statements

The following code example shows how to run batches of PartiQL statements on a DynamoDB table.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use batches of INSERT statements to add items.

```
///<summary>
///
///</summary>
///<param name="tableName"></param>
///<param name="title1"></param>
///<param name="title2"></param>
///<param name="year1"></param>
///<param name="year2"></param>
///<returns></returns>
public static async Task<bool> GetBatch(
    string tableName,
    string title1,
    string title2,
    int year1,
    int year2)
{
    var getBatch = $"SELECT FROM {tableName} WHERE title = ? AND year = ?";
    var statements = new List<BatchStatementRequest>
    {
        new BatchStatementRequest
        {
            Statement = getBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = title1 },
                new AttributeValue { N = year1.ToString() },
            },
            new BatchStatementRequest
            {
                Statement = getBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = title2 },
                    new AttributeValue { N = year2.ToString() },
                },
            }
        }
    };
    return await Client.BatchWriteItemAsync(tableName, statements);
}
```

```
    Parameters = new List<AttributeValue>
    {
        new AttributeValue { S = title2 },
        new AttributeValue { N = year2.ToString() },
    },
};

var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
{
    Statements = statements,
});

if (response.Responses.Count > 0)
{
    response.Responses.ForEach(r =>
    {
        Console.WriteLine($"{r.Item["title"]}\t{r.Item["year"]}");
    });
    return true;
}
else
{
    Console.WriteLine($"Couldn't find either {title1} or {title2}.");
    return false;
}

}
```

Use batches of SELECT statements to get items.

```
/// <summary>
/// Inserts movies imported from a JSON file into the movie table by
/// using an Amazon DynamoDB PartiQL INSERT statement.
/// </summary>
/// <param name="tableName">The name of the table into which the movie
/// information will be inserted.</param>
/// <param name="movieFileName">The name of the JSON file that contains
/// movie information.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the insert operation.</returns>
public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
{
    // Get the list of movies from the JSON file.
    var movies = ImportMovies(movieFileName);

    var success = false;

    if (movies is not null)
    {
        // Insert the movies in a batch using PartiQL. Because the
        // batch can contain a maximum of 25 items, insert 25 movies
        // at a time.
        string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?, 'year': ?}}";
        var statements = new List<BatchStatementRequest>();

        try
        {
            for (var indexOffset = 0; indexOffset < 250; indexOffset += 25)
```

```
{  
    for (var i = indexOffset; i < indexOffset + 25; i++)  
    {  
        statements.Add(new BatchStatementRequest  
        {  
            Statement = insertBatch,  
            Parameters = new List<AttributeValue>  
            {  
                new AttributeValue { S = movies[i].Title },  
                new AttributeValue { N =  
                    movies[i].Year.ToString() },  
            },  
        });  
    }  
  
    var response = await Client.BatchExecuteStatementAsync(new  
BatchExecuteStatementRequest  
    {  
        Statements = statements,  
    });  
  
    // Wait between batches for movies to be successfully added.  
    System.Threading.Thread.Sleep(3000);  
  
    success = response.HttpStatusCode ==  
System.Net.HttpStatusCode.OK;  
  
    // Clear the list of statements for the next batch.  
    statements.Clear();  
}  
}  
catch (AmazonDynamoDBException ex)  
{  
    Console.WriteLine(ex.Message);  
}  
}  
}  
  
return success;  
}  
  
/// <summary>  
/// Loads the contents of a JSON file into a list of movies to be  
/// added to the DynamoDB table.  
/// </summary>  
/// <param name="movieFileName">The full path to the JSON file.</param>  
/// <returns>A generic list of movie objects.</returns>  
public static List<Movie> ImportMovies(string movieFileName)  
{  
    if (!File.Exists(movieFileName))  
    {  
        return null;  
    }  
  
    using var sr = new StreamReader(movieFileName);  
    string json = sr.ReadToEnd();  
    var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);  
  
    if (allMovies is not null)  
    {  
        // Return the first 250 entries.  
        return allMovies.GetRange(0, 250);  
    }  
    else  
    {  
        return null;  
    }  
}
```

```
}
```

Use batches of UPDATE statements to update items.

```
/// <summary>
/// Updates information for multiple movies.
/// </summary>
/// <param name="tableName">The name of the table containing the
/// movies to be updated.</param>
/// <param name="producer1">The producer name for the first movie
/// to update.</param>
/// <param name="title1">The title of the first movie.</param>
/// <param name="year1">The year that the first movie was released.</param>
/// <param name="producer2">The producer name for the second
/// movie to update.</param>
/// <param name="title2">The title of the second movie.</param>
/// <param name="year2">The year that the second movie was released.</param>
/// <returns>A Boolean value that indicates the success of the update.</returns>
public static async Task<bool> UpdateBatch(
    string tableName,
    string producer1,
    string title1,
    int year1,
    string producer2,
    string title2,
    int year2)
{
    string updateBatch = $"UPDATE {tableName} SET Producer=? WHERE title = ?
AND year = ?";
    var statements = new List<BatchStatementRequest>
    {
        new BatchStatementRequest
        {
            Statement = updateBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = producer1 },
                new AttributeValue { S = title1 },
                new AttributeValue { N = year1.ToString() },
            },
        },
        new BatchStatementRequest
        {
            Statement = updateBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = producer2 },
                new AttributeValue { S = title2 },
                new AttributeValue { N = year2.ToString() },
            },
        }
    };

    var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
{
    Statements = statements,
});
}
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

Use batches of DELETE statements to delete items.

```
///<summary>
/// Deletes multiple movies using a PartiQL BatchExecuteAsync
/// statement.
///</summary>
///<param name="tableName">The name of the table containing the
/// moves that will be deleted.</param>
///<param name="title1">The title of the first movie.</param>
///<param name="year1">The year the first movie was released.</param>
///<param name="title2">The title of the second movie.</param>
///<param name="year2">The year the second movie was released.</param>
///<returns>A Boolean value indicating the success of the operation.</returns>
public static async Task<bool> DeleteBatch(
    string tableName,
    string title1,
    int year1,
    string title2,
    int year2)
{
    string updateBatch = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";
    var statements = new List<BatchStatementRequest>
    {
        new BatchStatementRequest
        {
            Statement = updateBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = title1 },
                new AttributeValue { N = year1.ToString() },
            },
        },
        new BatchStatementRequest
        {
            Statement = updateBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = title2 },
                new AttributeValue { N = year2.ToString() },
            },
        }
    };

    var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
{
    Statements = statements,
});

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for .NET API Reference*.

## Scan a table

The following code example shows how to scan a DynamoDB table.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static async Task<int> ScanTableAsync(
    AmazonDynamoDBClient client,
    string tableName,
    int startYear,
    int endYear)
{
    var request = new ScanRequest
    {
        TableName = tableName,
        ExpressionAttributeNames = new Dictionary<string, string>
        {
            { "#yr", "year" },
        },
        ExpressionAttributeValues = new Dictionary<string, AttributeValue>
        {
            { ":y_a", new AttributeValue { N = startYear.ToString() } },
            { ":y_z", new AttributeValue { N = endYear.ToString() } },
        },
        FilterExpression = "#yr between :y_a and :y_z",
        ProjectionExpression = "#yr, title, info.actors[0], info.directors,
info.running_time_secs",
    };

    // Keep track of how many movies were found.
    int foundCount = 0;

    var response = new ScanResponse();
    do
    {
        response = await client.ScanAsync(request);
        foundCount += response.Items.Count;
        response.Items.ForEach(i => DisplayItem(i));
    }
    while (response.LastEvaluatedKey.Count > 1);
    return foundCount;
}
```

- For API details, see [Scan in AWS SDK for .NET API Reference](#).

## Update an item in a table

The following code example shows how to update an item in a DynamoDB table.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    ///<summary>
    /// Updates an existing item in the movies table.
    ///</summary>
    ///<param name="client">An initialized Amazon DynamoDB client object.</param>
    ///<param name="newMovie">A Movie object containing information for
    /// the movie to update.</param>
    ///<param name="newInfo">A MovieInfo object that contains the
    /// information that will be changed.</param>
    ///<param name="tableName">The name of the table that contains the movie.</param>
    ///<returns>A Boolean value that indicates the success of the operation.</returns>
public static async Task<bool> UpdateItemAsync(
    AmazonDynamoDBClient client,
    Movie newMovie,
    MovieInfo newInfo,
    string tableName)
{
    var key = new Dictionary<string, AttributeValue>
    {
        ["title"] = new AttributeValue { S = newMovie.Title },
        ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
    };
    var updates = new Dictionary<string, AttributeValueUpdate>
    {
        ["info.plot"] = new AttributeValueUpdate
        {
            Action = AttributeAction.PUT,
            Value = new AttributeValue { S = newInfo.Plot },
        },
        ["info.rating"] = new AttributeValueUpdate
        {
            Action = AttributeAction.PUT,
            Value = new AttributeValue { N = newInfo.Rank.ToString() },
        },
    };

    var request = new UpdateItemRequest
    {
        AttributeUpdates = updates,
        Key = key,
        TableName = tableName,
    };

    var response = await client.UpdateItemAsync(request);

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [UpdateItem](#) in *AWS SDK for .NET API Reference*.

## Write a batch of items

The following code example shows how to write a batch of DynamoDB items.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Writes a batch of items to the movie table.

```
/// <summary>
/// Loads the contents of a JSON file into a list of movies to be
/// added to the DynamoDB table.
/// </summary>
/// <param name="movieFileName">The full path to the JSON file.</param>
/// <returns>A generic list of movie objects.</returns>
public static List<Movie> ImportMovies(string movieFileName)
{
    if (!File.Exists(movieFileName))
    {
        return null;
    }

    using var sr = new StreamReader(movieFileName);
    string json = sr.ReadToEnd();
    var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

    // Now return the first 250 entries.
    return allMovies.GetRange(0, 250);
}

/// <summary>
/// Writes 250 items to the movie table.
/// </summary>
/// <param name="client">The initialized DynamoDB client object.</param>
/// <param name="movieFileName">A string containing the full path to
/// the JSON file containing movie data.</param>
/// <returns>A long integer value representing the number of movies
/// imported from the JSON file.</returns>
public static async Task<long> BatchWriteItemsAsync(
    AmazonDynamoDBClient client,
    string movieFileName)
{
    var movies = ImportMovies(movieFileName);
    if (movies is null)
    {
        Console.WriteLine("Couldn't find the JSON file with movie data.");
        return 0;
    }

    var context = new DynamoDBContext(client);

    var bookBatch = context.CreateBatchWrite<Movie>();
    bookBatch.AddPutItems(movies);

    Console.WriteLine("Adding imported movies to the table.");
    await bookBatch.ExecuteAsync();

    return movies.Count;
}
```

- For API details, see [BatchWriteItem in AWS SDK for .NET API Reference](#).

## Scenarios

### Get started using tables, items, and queries

The following code example shows how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.
- Delete a movie from the table.
- Delete the table.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// This example application performs the following basic Amazon DynamoDB
// functions:
//
// CreateTableAsync
// PutItemAsync
// UpdateItemAsync
// BatchWriteItemAsync
// GetItemAsync
// DeleteItemAsync
// Query
// Scan
// DeleteItemAsync
//
// The code in this example uses the AWS SDK for .NET version 3.7 and .NET 5.
// Before you run this example, download 'movies.json' from
// https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files,
// and put it in the same folder as the example.
namespace DynamoDB_Basics_Scenario
{
    public class DynamoDB_Basics
    {
        // Separator for the console display.
        private static readonly string SepBar = new string('-', 80);

        public static async Task Main()
        {
            var client = new AmazonDynamoDBClient();

            var tableName = "movie_table";

            // relative path to moviedata.json in the local repository.
            var movieFileName = @"..\..\..\..\..\..\..\..\resources\sample_files
\movies.json";

            DisplayInstructions();

            // Create a new table and wait for it to be active.
            Console.WriteLine($"Creating the new table: {tableName}");

            var success = await DynamoDbMethods.CreateMovieTableAsync(client,
tableName);

            if (success)
            {
                Console.WriteLine($"\\nTable: {tableName} successfully created.");
            }
            else
```

```
{  
    Console.WriteLine($"\\nCould not create {tableName}.");  
}  
  
WaitForEnter();  
  
// Add a single new movie to the table.  
var newMovie = new Movie  
{  
    Year = 2021,  
    Title = "Spider-Man: No Way Home",  
};  
  
success = await DynamoDbMethods.PutItemAsync(client, newMovie, tableName);  
if (success)  
{  
    Console.WriteLine($"Added {newMovie.Title} to the table.");  
}  
else  
{  
    Console.WriteLine("Could not add movie to table.");  
}  
  
WaitForEnter();  
  
// Update the new movie by adding a plot and rank.  
var newInfo = new MovieInfo  
{  
    Plot = "With Spider-Man's identity now revealed, Peter asks" +  
        "Doctor Strange for help. When a spell goes wrong, dangerous" +  
        "foes from other worlds start to appear, forcing Peter to" +  
        "discover what it truly means to be Spider-Man.",  
    Rank = 9,  
};  
  
success = await DynamoDbMethods.UpdateItemAsync(client, newMovie, newInfo,  
tableName);  
if (success)  
{  
    Console.WriteLine($"Successfully updated the movie: {newMovie.Title}");  
}  
else  
{  
    Console.WriteLine("Could not update the movie.");  
}  
  
WaitForEnter();  
  
// Add a batch of movies to the DynamoDB table from a list of  
// movies in a JSON file.  
var itemCount = await DynamoDbMethods.BatchWriteItemsAsync(client,  
movieFileName);  
Console.WriteLine($"Added {itemCount} movies to the table.");  
  
WaitForEnter();  
  
// Get a movie by key. (partition + sort)  
var lookupMovie = new Movie  
{  
    Title = "Jurassic Park",  
    Year = 1993,  
};  
  
Console.WriteLine("Looking for the movie \\\"Jurassic Park\\\".");  
var item = await DynamoDbMethods.GetItemAsync(client, lookupMovie,  
tableName);
```

```
if (item.Count > 0)
{
    DynamoDbMethods.DisplayItem(item);
}
else
{
    Console.WriteLine($"Couldn't find {lookupMovie.Title}");
}

WaitForEnter();

// Delete a movie.
var movieToDelete = new Movie
{
    Title = "The Town",
    Year = 2010,
};

success = await DynamoDbMethods.DeleteItemAsync(client, tableName,
movieToDelete);

if (success)
{
    Console.WriteLine($"Successfully deleted {movieToDelete.Title}.");
}
else
{
    Console.WriteLine($"Could not delete {movieToDelete.Title}.");
}

WaitForEnter();

// Use Query to find all the movies released in 2010.
int findYear = 2010;
Console.WriteLine($"Movies released in {findYear}");
var queryCount = await DynamoDbMethods.QueryMoviesAsync(client, tableName,
findYear);
Console.WriteLine($"Found {queryCount} movies released in {findYear}");

WaitForEnter();

// Use Scan to get a list of movies from 2001 to 2011.
int startYear = 2001;
int endYear = 2011;
var scanCount = await DynamoDbMethods.ScanTableAsync(client, tableName,
startYear, endYear);
Console.WriteLine($"Found {scanCount} movies released between {startYear}
and {endYear}");

WaitForEnter();

// Delete the table.
success = await DynamoDbMethods.DeleteTableAsync(client, tableName);

if (success)
{
    Console.WriteLine($"Successfully deleted {tableName}");
}
else
{
    Console.WriteLine($"Could not delete {tableName}");
}

Console.WriteLine("The DynamoDB Basics example application is done.");

WaitForEnter();
```

```

    }

    ///<summary>
    /// Displays the description of the application on the console.
    ///</summary>
    private static void DisplayInstructions()
    {
        Console.Clear();
        Console.WriteLine();
        Console.Write(new string(' ', 28));
        Console.WriteLine("DynamoDB Basics Example");
        Console.WriteLine(SepBar);
        Console.WriteLine("This demo application shows the basics of using DynamoDB
with the AWS SDK for");
        Console.WriteLine(".NET version 3.7 and .NET Core 5.");
        Console.WriteLine(SepBar);
        Console.WriteLine("The application does the following:");
        Console.WriteLine("\t1. Creates a table with partition: year and
sort:title.");
        Console.WriteLine("\t2. Adds a single movie to the table.");
        Console.WriteLine("\t3. Adds movies to the table from moviedata.json.");
        Console.WriteLine("\t4. Updates the rating and plot of the movie that was
just added.");
        Console.WriteLine("\t5. Gets a movie using its key (partition + sort).");
        Console.WriteLine("\t6. Deletes a movie.");
        Console.WriteLine("\t7. Uses QueryAsync to return all movies released in a
given year.");
        Console.WriteLine("\t8. Uses ScanAsync to return all movies released within
a range of years.");
        Console.WriteLine("\t9. Finally, it deletes the table that was just
created.");
        WaitForEnter();
    }

    ///<summary>
    /// Simple method to wait for the Enter key to be pressed.
    ///</summary>
    private static void WaitForEnter()
    {
        Console.WriteLine("\nPress <Enter> to continue.");
        Console.WriteLine(SepBar);
        _ = Console.ReadLine();
    }
}
}

```

Creates a table to contain movie data.

```

    ///<summary>
    /// Creates a new Amazon DynamoDB table and then waits for the new
    /// table to become active.
    ///</summary>
    ///<param name="client">An initialized Amazon DynamoDB client object.</param>
    ///<param name="tableName">The name of the table to create.</param>
    ///<returns>A Boolean value indicating the success of the operation.</returns>
    public static async Task<bool> CreateMovieTableAsync(AmazonDynamoDBClient
client, string tableName)
    {
        var response = await client.CreateTableAsync(new CreateTableRequest
        {
            TableName = tableName,
            AttributeDefinitions = new List<AttributeDefinition>()
        });
    }
}

```

```
{  
    new AttributeDefinition  
    {  
        AttributeName = "title",  
        AttributeType = "S",  
    },  
    new AttributeDefinition  
    {  
        AttributeName = "year",  
        AttributeType = "N",  
    },  
},  
KeySchema = new List<KeySchemaElement>()  
{  
    new KeySchemaElement  
    {  
        AttributeName = "year",  
        KeyType = "HASH",  
    },  
    new KeySchemaElement  
    {  
        AttributeName = "title",  
        KeyType = "RANGE",  
    },  
},  
ProvisionedThroughput = new ProvisionedThroughput  
{  
    ReadCapacityUnits = 5,  
    WriteCapacityUnits = 5,  
},  
});  
  
// Wait until the table is ACTIVE and then report success.  
Console.WriteLine("Waiting for table to become active...");  
  
var request = new DescribeTableRequest  
{  
    TableName = response.TableDescription.TableName,  
};  
  
TableStatus status;  
  
int sleepDuration = 2000;  
  
do  
{  
    System.Threading.Thread.Sleep(sleepDuration);  
  
    var describeTableResponse = await client.DescribeTableAsync(request);  
    status = describeTableResponse.Table.TableStatus;  
  
    Console.WriteLine("...");  
}  
while (status != "ACTIVE");  
  
return status == TableStatus.ACTIVE;  
}
```

Adds a single movie to the table.

```
/// <summary>  
/// Adds a new item to the table.
```

```
    /// </summary>
    /// <param name="client">An initialized Amazon DynamoDB client object.</param>
    /// <param name="newMovie">A Movie object containing information for
    /// the movie to add to the table.</param>
    /// <param name="tableName">The name of the table where the item will be
    added.</param>
    /// <returns>A Boolean value that indicates the results of adding the item.</
    returns>
    public static async Task<bool> PutItemAsync(AmazonDynamoDBClient client, Movie
    newMovie, string tableName)
    {
        var item = new Dictionary<string, AttributeValue>
        {
            ["title"] = new AttributeValue { S = newMovie.Title },
            ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
        };

        var request = new PutItemRequest
        {
            TableName = tableName,
            Item = item,
        };

        var response = await client.PutItemAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

Updates a single item in a table.

```
    /// <summary>
    /// Updates an existing item in the movies table.
    /// </summary>
    /// <param name="client">An initialized Amazon DynamoDB client object.</param>
    /// <param name="newMovie">A Movie object containing information for
    /// the movie to update.</param>
    /// <param name="newInfo">A MovieInfo object that contains the
    /// information that will be changed.</param>
    /// <param name="tableName">The name of the table that contains the movie.</
    param>
    /// <returns>A Boolean value that indicates the success of the operation.</
    returns>
    public static async Task<bool> UpdateItemAsync(
        AmazonDynamoDBClient client,
        Movie newMovie,
        MovieInfo newInfo,
        string tableName)
    {
        var key = new Dictionary<string, AttributeValue>
        {
            ["title"] = new AttributeValue { S = newMovie.Title },
            ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
        };
        var updates = new Dictionary<string, AttributeValueUpdate>
        {
            ["info.plot"] = new AttributeValueUpdate
            {
                Action = AttributeAction.PUT,
                Value = new AttributeValue { S = newInfo.Plot },
            },
            ["info.rating"] = new AttributeValueUpdate
            {
```

```
        Action = AttributeAction.PUT,
        Value = new AttributeValue { N = newInfo.Rank.ToString() },
    },
};

var request = new UpdateItemRequest
{
    AttributeUpdates = updates,
    Key = key,
    TableName = tableName,
};

var response = await client.UpdateItemAsync(request);

return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

Retrieves a single item from the movie table.

```
/// <summary>
/// Gets information about an existing movie from the table.
/// </summary>
/// <param name="client">An initialized Amazon DynamoDB client object.</param>
/// <param name="newMovie">A Movie object containing information about
/// the movie to retrieve.</param>
/// <param name="tableName">The name of the table containing the movie.</param>
/// <returns>A Dictionary object containing information about the item
/// retrieved.</returns>
public static async Task<Dictionary<string, AttributeValue>>
GetItemAsync(AmazonDynamoDBClient client, Movie newMovie, string tableName)
{
    var key = new Dictionary<string, AttributeValue>
    {
        ["title"] = new AttributeValue { S = newMovie.Title },
        ["year"] = new AttributeValue { N = newMovie.Year.ToString() },
    };

    var request = new GetItemRequest
    {
        Key = key,
        TableName = tableName,
    };

    var response = await client.GetItemAsync(request);
    return response.Item;
}
```

Writes a batch of items to the movie table.

```
/// <summary>
/// Loads the contents of a JSON file into a list of movies to be
/// added to the DynamoDB table.
/// </summary>
/// <param name="movieFileName">The full path to the JSON file.</param>
/// <returns>A generic list of movie objects.</returns>
public static List<Movie> ImportMovies(string movieFileName)
{
```

```

        if (!File.Exists(movieFileName))
        {
            return null;
        }

        using var sr = new StreamReader(movieFileName);
        string json = sr.ReadToEnd();
        var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

        // Now return the first 250 entries.
        return allMovies.GetRange(0, 250);
    }

    ///<summary>
    /// Writes 250 items to the movie table.
    ///</summary>
    ///<param name="client">The initialized DynamoDB client object.</param>
    ///<param name="movieFileName">A string containing the full path to
    ///the JSON file containing movie data.</param>
    ///<returns>A long integer value representing the number of movies
    ///imported from the JSON file.</returns>
    public static async Task<long> BatchWriteItemsAsync(
        AmazonDynamoDBClient client,
        string movieFileName)
    {
        var movies = ImportMovies(movieFileName);
        if (movies is null)
        {
            Console.WriteLine("Couldn't find the JSON file with movie data.");
            return 0;
        }

        var context = new DynamoDBContext(client);

        var bookBatch = context.CreateBatchWrite<Movie>();
        bookBatch.AddPutItems(movies);

        Console.WriteLine("Adding imported movies to the table.");
        await bookBatch.ExecuteAsync();

        return movies.Count;
    }
}

```

Deletes a single item from the table.

```

    ///<summary>
    /// Deletes a single item from a DynamoDB table.
    ///</summary>
    ///<param name="client">The initialized DynamoDB client object.</param>
    ///<param name="tableName">The name of the table from which the item
    ///will be deleted.</param>
    ///<param name="movieToDelete">A movie object containing the title and
    ///year of the movie to delete.</param>
    ///<returns>A Boolean value indicating the success or failure of the
    ///delete operation.</returns>
    public static async Task<bool> DeleteItemAsync(
        AmazonDynamoDBClient client,
        string tableName,
        Movie movieToDelete)
    {
        var key = new Dictionary<string, AttributeValue>
        {

```

```
        ["title"] = new AttributeValue { S = movieToDelete.Title },
        ["year"] = new AttributeValue { N = movieToDelete.Year.ToString() },
    };

    var request = new DeleteItemRequest
    {
        TableName = tableName,
        Key = key,
    };

    var response = await client.DeleteItemAsync(request);
    return response.HttpStatusCode == System.Net HttpStatusCode.OK;
}
```

Queries the table for movies released in a particular year.

```
/// <summary>
/// Queries the table for movies released in a particular year and
/// then displays the information for the movies returned.
/// </summary>
/// <param name="client">The initialized DynamoDB client object.</param>
/// <param name="tableName">The name of the table to query.</param>
/// <param name="year">The release year for which we want to
/// view movies.</param>
/// <returns>The number of movies that match the query.</returns>
public static async Task<int> QueryMoviesAsync(AmazonDynamoDBClient client,
string tableName, int year)
{
    var movieTable = Table.LoadTable(client, tableName);
    var filter = new QueryFilter("year", QueryOperator.Equal, year);

    Console.WriteLine("\nFind movies released in: {year}:");

    var config = new QueryOperationConfig()
    {
        Limit = 10, // 10 items per page.
        Select = SelectValues.SpecificAttributes,
        AttributesToGet = new List<string>
        {
            "title",
            "year",
        },
        ConsistentRead = true,
        Filter = filter,
    };

    // Value used to track how many movies match the
    // supplied criteria.
    var moviesFound = 0;

    Search search = movieTable.Query(config);
    do
    {
        var movieList = await search.GetNextSetAsync();
        moviesFound += movieList.Count;

        foreach (var movie in movieList)
        {
            DisplayDocument(movie);
        }
    }
    while (!search.IsDone);
```

```
        return moviesFound;
    }
```

Scans the table for movies released in a range of years.

```
public static async Task<int> ScanTableAsync(
    AmazonDynamoDBClient client,
    string tableName,
    int startYear,
    int endYear)
{
    var request = new ScanRequest
    {
        TableName = tableName,
        ExpressionAttributeNames = new Dictionary<string, string>
        {
            { "#yr", "year" },
        },
        ExpressionAttributeValues = new Dictionary<string, AttributeValue>
        {
            { ":y_a", new AttributeValue { N = startYear.ToString() } },
            { ":y_z", new AttributeValue { N = endYear.ToString() } },
        },
        FilterExpression = "#yr between :y_a and :y_z",
        ProjectionExpression = "#yr, title, info.actors[0], info.directors,
info.running_time_secs",
    };

    // Keep track of how many movies were found.
    int foundCount = 0;

    var response = new ScanResponse();
    do
    {
        response = await client.ScanAsync(request);
        foundCount += response.Items.Count;
        response.Items.ForEach(i => DisplayItem(i));
    }
    while (response.LastEvaluatedKey.Count > 1);
    return foundCount;
}
```

Deletes the movie table.

```
public static async Task<bool> DeleteTableAsync(AmazonDynamoDBClient client,
string tableName)
{
    var request = new DeleteTableRequest
    {
        TableName = tableName,
    };

    var response = await client.DeleteTableAsync(request);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"Table {response.TableDescription.TableName}
successfully deleted.");
        return true;
    }
}
```

```
        }
    else
    {
        Console.WriteLine("Could not delete table.");
        return false;
    }
}
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

### Query a table by using batches of PartiQL statements

The following code example shows how to:

- Get a batch of items by running multiple SELECT statements.
- Add a batch of items by running multiple INSERT statements.
- Update a batch of items by running multiple UPDATE statements.
- Delete a batch of items by running multiple DELETE statements.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Before you run this example, download 'movies.json' from
// https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/
GettingStarted.Js.02.html,
// and put it in the same folder as the example.

// Separator for the console display.
var SepBar = new string('-', 80);
const string tableName = "movie_table";
const string movieFileName = "moviedata.json";

DisplayInstructions();

// Create the table and wait for it to be active.
Console.WriteLine($"Creating the movie table: {tableName}");

var success = await DynamoDBMethods.CreateMovieTableAsync(tableName);
if (success)
```

```
{  
    Console.WriteLine($"Successfully created table: {tableName}.");  
}  
  
WaitForEnter();  
  
// Add movie information to the table from moviedata.json. See the  
// instructions at the top of this file to download the JSON file.  
Console.WriteLine($"Inserting movies into the new table. Please wait...");  
success = await PartiQLBatchMethods.InsertMovies(tableName, movieFileName);  
if (success)  
{  
    Console.WriteLine("Movies successfully added to the table.");  
}  
else  
{  
    Console.WriteLine("Movies could not be added to the table.");  
}  
  
WaitForEnter();  
  
// Update multiple movies by using the BatchExecute statement.  
var title1 = "Star Wars";  
var year1 = 1977;  
var title2 = "Wizard of Oz";  
var year2 = 1939;  
  
Console.WriteLine($"Updating two movies with producer information: {title1} and  
{title2}.");  
success = await PartiQLBatchMethods.GetBatch(tableName, title1, title2, year1, year2);  
if (success)  
{  
    Console.WriteLine($"Successfully retrieved {title1} and {title2}.");  
}  
else  
{  
    Console.WriteLine("Select statement failed.");  
}  
  
WaitForEnter();  
  
// Update multiple movies by using the BatchExecute statement.  
var producer1 = "LucasFilm";  
var producer2 = "MGM";  
  
Console.WriteLine($"Updating two movies with producer information: {title1} and  
{title2}.");  
success = await PartiQLBatchMethods.UpdateBatch(tableName, producer1, title1, year1,  
producer2, title2, year2);  
if (success)  
{  
    Console.WriteLine($"Successfully updated {title1} and {title2}.");  
}  
else  
{  
    Console.WriteLine("Update failed.");  
}  
  
WaitForEnter();  
  
// Delete multiple movies by using the BatchExecute statement.  
Console.WriteLine($"Now we will delete {title1} and {title2} from the table.");  
success = await PartiQLBatchMethods.DeleteBatch(tableName, title1, year1, title2,  
year2);  
  
if (success)
```

```
{  
    Console.WriteLine($"Deleted {title1} and {title2}");  
}  
else  
{  
    Console.WriteLine($"could not delete {title1} or {title2}");  
}  
  
WaitForEnter();  
  
// DNow that the PartiQL Batch scenario is complete, delete the movie table.  
success = await DynamoDBMethods.DeleteTableAsync(tableName);  
  
if (success)  
{  
    Console.WriteLine($"Successfully deleted {tableName}");  
}  
else  
{  
    Console.WriteLine($"Could not delete {tableName}");  
}  
  
/// <summary>  
/// Displays the description of the application on the console.  
/// </summary>  
void DisplayInstructions()  
{  
    Console.Clear();  
    Console.WriteLine();  
    Console.Write(new string(' ', 24));  
    Console.WriteLine("DynamoDB PartiQL Basics Example");  
    Console.WriteLine(SepBar);  
    Console.WriteLine("This demo application shows the basics of using Amazon DynamoDB  
with the AWS SDK for");  
    Console.WriteLine(".NET version 3.7 and .NET 6.");  
    Console.WriteLine(SepBar);  
    Console.WriteLine("Creates a table by using the CreateTable method.");  
    Console.WriteLine("Gets multiple movies by using a PartiQL SELECT statement.");  
    Console.WriteLine("Updates multiple movies by using the ExecuteBatch method.");  
    Console.WriteLine("Deletes multiple movies by using a PartiQL DELETE statement.");  
    Console.WriteLine("Cleans up the resources created for the demo by deleting the  
table.");  
    Console.WriteLine(SepBar);  
  
    WaitForEnter();  
}  
  
/// <summary>  
/// Simple method to wait for the <Enter> key to be pressed.  
/// </summary>  
void WaitForEnter()  
{  
    Console.WriteLine("\nPress <Enter> to continue.");  
    Console.Write(SepBar);  
    _ = Console.ReadLine();  
}  
  
/// <summary>  
///  
/// </summary>  
/// <param name="tableName"></param>  
/// <param name="title1"></param>  
/// <param name="title2"></param>  
/// <param name="year1"></param>
```

```

    ///> <param name="year2"></param>
    ///> <returns></returns>
    public static async Task<bool> GetBatch(
        string tableName,
        string title1,
        string title2,
        int year1,
        int year2)
    {
        var getBatch = $"SELECT FROM {tableName} WHERE title = ? AND year = ?";
        var statements = new List<BatchStatementRequest>
        {
            new BatchStatementRequest
            {
                Statement = getBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = title1 },
                    new AttributeValue { N = year1.ToString() },
                },
            },
            new BatchStatementRequest
            {
                Statement = getBatch,
                Parameters = new List<AttributeValue>
                {
                    new AttributeValue { S = title2 },
                    new AttributeValue { N = year2.ToString() },
                },
            }
        };

        var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
{
    Statements = statements,
});

        if (response.Responses.Count > 0)
        {
            response.Responses.ForEach(r =>
            {
                Console.WriteLine($"{r.Item["title"]}\t{r.Item["year"]}");
            });
            return true;
        }
        else
        {
            Console.WriteLine($"Couldn't find either {title1} or {title2}.");
            return false;
        }
    }

    ///> <summary>
    ///> Inserts movies imported from a JSON file into the movie table by
    ///> using an Amazon DynamoDB PartiQL INSERT statement.
    ///> </summary>
    ///> <param name="tableName">The name of the table into which the movie
    ///> information will be inserted.</param>
    ///> <param name="movieFileName">The name of the JSON file that contains
    ///> movie information.</param>
    ///> <returns>A Boolean value that indicates the success or failure of

```

```

    ///> the insert operation.</returns>
    public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
    {
        // Get the list of movies from the JSON file.
        var movies = ImportMovies(movieFileName);

        var success = false;

        if (movies is not null)
        {
            // Insert the movies in a batch using PartiQL. Because the
            // batch can contain a maximum of 25 items, insert 25 movies
            // at a time.
            string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?, 'year': ?}}";
            var statements = new List<BatchStatementRequest>();

            try
            {
                for (var indexOffset = 0; indexOffset < 250; indexOffset += 25)
                {
                    for (var i = indexOffset; i < indexOffset + 25; i++)
                    {
                        statements.Add(new BatchStatementRequest
                        {
                            Statement = insertBatch,
                            Parameters = new List<AttributeValue>
                            {
                                new AttributeValue { S = movies[i].Title },
                                new AttributeValue { N = movies[i].Year.ToString() }
                            },
                        });
                    }
                }

                var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
                {
                    Statements = statements,
                });

                // Wait between batches for movies to be successfully added.
                System.Threading.Thread.Sleep(3000);

                success = response.HttpStatusCode ==
System.Net.HttpStatusCode.OK;

                // Clear the list of statements for the next batch.
                statements.Clear();
            }
        }
        catch (AmazonDynamoDBException ex)
        {
            Console.WriteLine(ex.Message);
        }
    }

    return success;
}

///<summary>
/// Loads the contents of a JSON file into a list of movies to be
/// added to the DynamoDB table.
///</summary>
///<param name="movieFileName">The full path to the JSON file.</param>

```

```

/// <returns>A generic list of movie objects.</returns>
public static List<Movie> ImportMovies(string movieFileName)
{
    if (!File.Exists(movieFileName))
    {
        return null;
    }

    using var sr = new StreamReader(movieFileName);
    string json = sr.ReadToEnd();
    var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

    if (allMovies is not null)
    {
        // Return the first 250 entries.
        return allMovies.GetRange(0, 250);
    }
    else
    {
        return null;
    }
}

/// <summary>
/// Updates information for multiple movies.
/// </summary>
/// <param name="tableName">The name of the table containing the
/// movies to be updated.</param>
/// <param name="producer1">The producer name for the first movie
/// to update.</param>
/// <param name="title1">The title of the first movie.</param>
/// <param name="year1">The year that the first movie was released.</param>
/// <param name="producer2">The producer name for the second
/// movie to update.</param>
/// <param name="title2">The title of the second movie.</param>
/// <param name="year2">The year that the second movie was released.</param>
/// <returns>A Boolean value that indicates the success of the update.</
returns>
public static async Task<bool> UpdateBatch(
    string tableName,
    string producer1,
    string title1,
    int year1,
    string producer2,
    string title2,
    int year2)
{

    string updateBatch = $"UPDATE {tableName} SET Producer=? WHERE title = ?
AND year = ?";
    var statements = new List<BatchStatementRequest>
    {
        new BatchStatementRequest
        {
            Statement = updateBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = producer1 },
                new AttributeValue { S = title1 },
                new AttributeValue { N = year1.ToString() },
            },
        },
        new BatchStatementRequest
    };
}

```

```
        {
            Statement = updateBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = producer2 },
                new AttributeValue { S = title2 },
                new AttributeValue { N = year2.ToString() },
            },
        },
    };

    var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
{
    Statements = statements,
});

return response.StatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Deletes multiple movies using a PartiQL BatchExecuteAsync
/// statement.
/// </summary>
/// <param name="tableName">The name of the table containing the
/// moves that will be deleted.</param>
/// <param name="title1">The title of the first movie.</param>
/// <param name="year1">The year the first movie was released.</param>
/// <param name="title2">The title of the second movie.</param>
/// <param name="year2">The year the second movie was released.</param>
/// <returns>A Boolean value indicating the success of the operation.</returns>
public static async Task<bool> DeleteBatch(
    string tableName,
    string title1,
    int year1,
    string title2,
    int year2)
{
    string updateBatch = $"DELETE FROM {tableName} WHERE title = ? AND year
= ?";
    var statements = new List<BatchStatementRequest>
    {
        new BatchStatementRequest
        {
            Statement = updateBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = title1 },
                new AttributeValue { N = year1.ToString() },
            },
        },
        new BatchStatementRequest
        {
            Statement = updateBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = title2 },
                new AttributeValue { N = year2.ToString() },
            },
        }
    };
}
```

```
        var response = await Client.BatchExecuteStatementAsync(new
BatchExecuteStatementRequest
{
    Statements = statements,
});

return response.StatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for .NET API Reference*.

## Query a table using PartiQL

The following code example shows how to:

- Get an item by running a SELECT statement.
- Add an item by running an INSERT statement.
- Update an item by running an UPDATE statement.
- Delete an item by running a DELETE statement.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace PartiQL_Basics_Scenario
{
    public class PartiQLMethods
    {
        private static readonly AmazonDynamoDBClient Client = new
AmazonDynamoDBClient();

        ///<summary>
        /// Inserts movies imported from a JSON file into the movie table by
        /// using an Amazon DynamoDB PartiQL INSERT statement.
        /// </summary>
        /// <param name="tableName">The name of the table where the movie
        /// information will be inserted.</param>
        /// <param name="movieFileName">The name of the JSON file that contains
        /// movie information.</param>
        /// <returns>A Boolean value that indicates the success or failure of
        /// the insert operation.</returns>
        public static async Task<bool> InsertMovies(string tableName, string
movieFileName)
        {
            // Get the list of movies from the JSON file.
            var movies = ImportMovies(movieFileName);

            var success = false;

            if (movies is not null)
            {
                // Insert the movies in a batch using PartiQL. Because the
                // batch can contain a maximum of 25 items, insert 25 movies
                // at a time.
            }
        }
    }
}
```

```
        string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?,  
'year': ?}}";  
        var statements = new List<BatchStatementRequest>();  
  
        try  
        {  
            for (var indexOffset = 0; indexOffset < 250; indexOffset += 25)  
            {  
                for (var i = indexOffset; i < indexOffset + 25; i++)  
                {  
                    statements.Add(new BatchStatementRequest  
                    {  
                        Statement = insertBatch,  
                        Parameters = new List<AttributeValue>  
                        {  
                            new AttributeValue { S = movies[i].Title },  
                            new AttributeValue { N =  
movies[i].Year.ToString() },  
                        },  
                    });  
                }  
            }  
  
            var response = await Client.BatchExecuteStatementAsync(new  
BatchExecuteStatementRequest  
            {  
                Statements = statements,  
            });  
  
            // Wait between batches for movies to be successfully added.  
            System.Threading.Thread.Sleep(3000);  
  
            success = response.HttpStatusCode ==  
System.Net.HttpStatusCode.OK;  
  
            // Clear the list of statements for the next batch.  
            statements.Clear();  
        }  
        catch (AmazonDynamoDBException ex)  
        {  
            Console.WriteLine(ex.Message);  
        }  
    }  
  
    return success;  
}  
  
/// <summary>  
/// Loads the contents of a JSON file into a list of movies to be  
/// added to the DynamoDB table.  
/// </summary>  
/// <param name="movieFileName">The full path to the JSON file.</param>  
/// <returns>A generic list of movie objects.</returns>  
public static List<Movie> ImportMovies(string movieFileName)  
{  
    if (!File.Exists(movieFileName))  
    {  
        return null;  
    }  
  
    using var sr = new StreamReader(movieFileName);  
    string json = sr.ReadToEnd();  
    var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);  
  
    if (allMovies is not null)  
    {
```

```
// Return the first 250 entries.  
    return allMovies.GetRange(0, 250);  
}  
else  
{  
    return null;  
}  
}  
  
  
/// <summary>  
/// Uses a PartiQL SELECT statement to retrieve a single movie from the  
/// movie database.  
/// </summary>  
/// <param name="tableName">The name of the movie table.</param>  
/// <param name="movieTitle">The title of the movie to retrieve.</param>  
/// <returns>A list of movie data. If no movie matches the supplied  
/// title, the list is empty.</returns>  
public static async Task<List<Dictionary<string,AttributeValue>>>  
GetSingleMovie(string tableName, string movieTitle)  
{  
    string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";  
    var parameters = new List<AttributeValue>  
    {  
        new AttributeValue { S = movieTitle },  
    };  
  
    var response = await Client.ExecuteStatementAsync(new  
ExecuteStatementRequest  
    {  
        Statement = selectSingle,  
        Parameters = parameters,  
    });  
  
    return response.Items;  
}  
  
  
/// <summary>  
/// Retrieve multiple movies by year using a SELECT statement.  
/// </summary>  
/// <param name="tableName">The name of the movie table.</param>  
/// <param name="year">The year the movies were released.</param>  
/// <returns></returns>  
public static async Task<List<Dictionary<string,AttributeValue>>>  
GetMovies(string tableName, int year)  
{  
    string selectSingle = $"SELECT * FROM {tableName} WHERE year = ?";  
    var parameters = new List<AttributeValue>  
    {  
        new AttributeValue { N = year.ToString() },  
    };  
  
    var response = await Client.ExecuteStatementAsync(new  
ExecuteStatementRequest  
    {  
        Statement = selectSingle,  
        Parameters = parameters,  
    });  
  
    return response.Items;  
}
```

```

    ///<summary>
    /// Inserts a single movie into the movies table.
    ///</summary>
    ///<param name="tableName">The name of the table.</param>
    ///<param name="movieTitle">The title of the movie to insert.</param>
    ///<param name="year">The year that the movie was released.</param>
    ///<returns>A Boolean value that indicates the success or failure of
    /// the INSERT operation.</returns>
    public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
    {
        string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?, 'year': ?}}";

        var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
        {
            Statement = insertBatch,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = movieTitle },
                new AttributeValue { N = year.ToString() },
            },
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    ///<summary>
    /// Updates a single movie in the table, adding information for the
    /// producer.
    ///</summary>
    ///<param name="tableName">the name of the table.</param>
    ///<param name="producer">The name of the producer.</param>
    ///<param name="movieTitle">The movie title.</param>
    ///<param name="year">The year the movie was released.</param>
    ///<returns>A Boolean value that indicates the success of the
    /// UPDATE operation.</returns>
    public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
    {
        string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title = ?
AND year = ?";

        var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
        {
            Statement = insertSingle,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = producer },
                new AttributeValue { S = movieTitle },
                new AttributeValue { N = year.ToString() },
            },
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    ///<summary>
    /// Deletes a single movie from the table.
    ///</summary>

```

```

    ///> <param name="tableName">The name of the table.</param>
    ///> <param name="movieTitle">The title of the movie to delete.</param>
    ///> <param name="year">The year that the movie was released.</param>
    ///> <returns>A Boolean value that indicates the success of the
    ///> DELETE operation.</returns>
    public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
    {
        var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year = ?";

        var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
        {
            Statement = deleteSingle,
            Parameters = new List<AttributeValue>
            {
                new AttributeValue { S = movieTitle },
                new AttributeValue { N = year.ToString() },
            },
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    ///> <summary>
    ///> Displays the list of movies returned from a database query.
    ///> </summary>
    ///> <param name="items">The list of movie information to display.</param>
    private static void DisplayMovies(List<Dictionary<string,AttributeValue>>
items)
    {
        if (items.Count > 0)
        {
            Console.WriteLine($"Found {items.Count} movies.");
            items.ForEach(item =>
Console.WriteLine($"{item["year"]}\t{item["title"]}");
        }
        else
        {
            Console.WriteLine($"Didn't find a movie that matched the supplied
criteria.");
        }
    }
}

    ///> <summary>
    ///> Uses a PartiQL SELECT statement to retrieve a single movie from the
    ///> movie database.
    ///> </summary>
    ///> <param name="tableName">The name of the movie table.</param>
    ///> <param name="movieTitle">The title of the movie to retrieve.</param>
    ///> <returns>A list of movie data. If no movie matches the supplied
    ///> title, the list is empty.</returns>
    public static async Task<List<Dictionary<string,AttributeValue>>>
GetSingleMovie(string tableName, string movieTitle)
    {
        string selectSingle = $"SELECT * FROM {tableName} WHERE title = ?";
        var parameters = new List<AttributeValue>
        {
            new AttributeValue { S = movieTitle },

```

```

    };

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = selectSingle,
    Parameters = parameters,
});

return response.Items;
}

/// <summary>
/// Inserts a single movie into the movies table.
/// </summary>
/// <param name="tableName">The name of the table.</param>
/// <param name="movieTitle">The title of the movie to insert.</param>
/// <param name="year">The year that the movie was released.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the INSERT operation.</returns>
public static async Task<bool> InsertSingleMovie(string tableName, string
movieTitle, int year)
{
    string insertBatch = $"INSERT INTO {tableName} VALUE {{'title': ?, 'year': ?}}";

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = insertBatch,
    Parameters = new List<AttributeValue>
    {
        new AttributeValue { S = movieTitle },
        new AttributeValue { N = year.ToString() },
    },
});

return response.StatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Updates a single movie in the table, adding information for the
/// producer.
/// </summary>
/// <param name="tableName">the name of the table.</param>
/// <param name="producer">The name of the producer.</param>
/// <param name="movieTitle">The movie title.</param>
/// <param name="year">The year the movie was released.</param>
/// <returns>A Boolean value that indicates the success of the
/// UPDATE operation.</returns>
public static async Task<bool> UpdateSingleMovie(string tableName, string
producer, string movieTitle, int year)
{
    string insertSingle = $"UPDATE {tableName} SET Producer=? WHERE title = ? AND year = ?";

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = insertSingle,
    Parameters = new List<AttributeValue>
    {
}

```

```
        new AttributeValue { S = producer },
        new AttributeValue { S = movieTitle },
        new AttributeValue { N = year.ToString() },
    },
});

return response.StatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Deletes a single movie from the table.
/// </summary>
/// <param name="tableName">The name of the table.</param>
/// <param name="movieTitle">The title of the movie to delete.</param>
/// <param name="year">The year that the movie was released.</param>
/// <returns>A Boolean value that indicates the success of the
/// DELETE operation.</returns>
public static async Task<bool> DeleteSingleMovie(string tableName, string
movieTitle, int year)
{
    var deleteSingle = $"DELETE FROM {tableName} WHERE title = ? AND year = ?";

    var response = await Client.ExecuteStatementAsync(new
ExecuteStatementRequest
{
    Statement = deleteSingle,
    Parameters = new List<AttributeValue>
    {
        new AttributeValue { S = movieTitle },
        new AttributeValue { N = year.ToString() },
    },
});

return response.StatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for .NET API Reference*.

## Amazon EC2 examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon Elastic Compute Cloud.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2344\)](#)

## Actions

### Create a VPC

The following code example shows how to create an Amazon EC2 VPC.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to create an Amazon Elastic Compute Cloud (Amazon EC2) VPC
/// using the AWS SDK for .NET and .NET Core 5.0.
/// </summary>
public class CreateVPC
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and then calls the
    /// CreateVpcAsync method to create the VPC.
    /// </summary>
    public static async Task Main()
    {
        // If you do not want to create the VPC in the same AWS Region as
        // the default users on your system, you need to supply the AWS
        // Region as a parameter to the client constructor.
        var client = new AmazonEC2Client();

        var response = await client.CreateVpcAsync(new CreateVpcRequest
        {
            CidrBlock = "10.0.0.0/16",
        });

        Vpc vpc = response.Vpc;

        if (vpc is not null)
        {
            Console.WriteLine($"Created VPC with ID: {vpc.VpcId}.");
        }
    }
}
```

- For API details, see [CreateVpc](#) in *AWS SDK for .NET API Reference*.

## Create a security group

The following code example shows how to create an Amazon EC2 security group.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;
```

```
/// <summary>
/// Shows how to create a security group for an Amazon Elastic Compute
/// Cloud (Amazon EC2) VPC using the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
/// </summary>
public class CreateSecurityGroup
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and uses the
    /// CreateSecurityGroupAsync method to create the security group.
    /// </summary>
    public static async Task Main()
    {
        string vpcId = "vpc-1234567890abcdefa";
        string vpcDescription = "Sample security group";
        string groupName = "sample-security-group";

        var client = new AmazonEC2Client();
        var response = await client.CreateSecurityGroupAsync(new
CreateSecurityGroupRequest
        {
            Description = vpcDescription,
           GroupName = groupName,
            VpcId = vpcId,
        });

        string groupId = response.GroupId;

        Console.WriteLine($"Successfully created security group: {groupName} with
ID: {groupId}");
    }
}
```

- For API details, see [CreateSecurityGroup](#) in *AWS SDK for .NET API Reference*.

## Create a security key pair

The following code example shows how to create a security key pair for Amazon EC2.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to create a new Amazon Elastic Compute Cloud (Amazon EC2)
/// key pair. The example was uses the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
/// </summary>
public class CreateKeyPair
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and then calls the
    /// CreateKeyPairAsync method to create the new key pair.
}
```

```
/// </summary>
public static async Task Main()
{
    string keyName = "sdk-example-key-pair";

    // If the default user on your system is not the same as
    // the Region where you want to create the key pair, you
    // need to supply the AWS Region as a parameter to the
    // client constructor.
    var client = new AmazonEC2Client();

    var request = new CreateKeyPairRequest
    {
        KeyName = keyName,
    };

    var response = await client.CreateKeyPairAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        var kp = response.KeyPair;
        Console.WriteLine($"{kp.KeyName} with the ID: {kp.KeyPairId}.");
    }
    else
    {
        Console.WriteLine("Could not create key pair.");
    }
}
```

- For API details, see [CreateKeyPair](#) in *AWS SDK for .NET API Reference*.

## Delete a VPC

The following code example shows how to delete an Amazon EC2 VPC.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to delete an existing Amazon Elastic Compute Cloud
/// (Amazon EC2) VPC. The example was created using the AWS SDK for .NET
/// version 3.7 and .NET Core 5.0.
/// </summary>
public class DeleteVPC
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and then calls the
    /// DeleteVpcAsync method to delete the VPC.
    /// </summary>
    public static async Task Main()
    {
```

```
string vpcId = "vpc-0123456789abc";

// If your Amazon EC2 VPC is not defined in the same AWS Region as
// the default AWS user on your system, you need to supply the AWS
// Region as a parameter to the client constructor.
var client = new AmazonEC2Client();

var request = new DeleteVpcRequest
{
    VpcId = vpcId,
};

var response = await client.DeleteVpcAsync(request);

if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
{
    Console.WriteLine($"Successfully deleted VPC with ID: {vpcId}.");
}
}
```

- For API details, see [DeleteVpc](#) in *AWS SDK for .NET API Reference*.

## Delete a security group

The following code example shows how to delete an Amazon EC2 security group.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to use Amazon Elastic Compute Cloud (Amazon EC2) and the
/// AWS SDK for .NET to delete an existing security group. This example
/// was created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DeleteSecurityGroup
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and then uses it to call
    /// the DeleteSecurityGroupAsync method to delete the security group.
    /// </summary>
    public static async Task Main()
    {
        string secGroupId = "sg-1234567890abcd";
        string groupName = "sample-security-group";

        // If your Amazon EC2 security group is not defined in the same AWS
        // Region as the default user on your system, you need to supply
        // the AWS Region as a parameter to the client constructor.
        var client = new AmazonEC2Client();

        var request = new DeleteSecurityGroupRequest
        {
```

```
        GroupId = secGroupId,
        GroupName = groupName,
    };

    var response = await client.DeleteSecurityGroupAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"Successfully deleted {groupName}.");
    }
    else
    {
        Console.WriteLine($"Could not delete {groupName}.");
    }
}
```

- For API details, see [DeleteSecurityGroup](#) in *AWS SDK for .NET API Reference*.

## Delete a security key pair

The following code example shows how to delete an Amazon EC2 security key pair.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to delete an existing Amazon Elastic Compute Cloud
/// (Amazon EC2) key pair. The example uses the AWS SDK for .NET
/// version 3.7 and .NET Core 5.0.
/// </summary>
public class DeleteKeyPair
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and then calls the
    /// DeleteKeyPairAsync method to delete the key pair.
    /// </summary>
    public static async Task Main()
    {
        string keyName = "sdk-example-key-pair";

        // If the key pair was not created in the same AWS Region as
        // the default user on your system, you need to supply
        // the AWS Region as a parameter to the client constructor.
        var client = new AmazonEC2Client();

        var request = new DeleteKeyPairRequest
        {
            KeyName = keyName,
        };

        var response = await client.DeleteKeyPairAsync(request);
```

```
        if (response.StatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"Successfully deleted the key pair: {keyName}.");
    }
    else
    {
        Console.WriteLine("Could not delete the key pair.");
    }
}
```

- For API details, see [DeleteKeyPair](#) in *AWS SDK for .NET API Reference*.

## Describe instances

The following code example shows how to describe Amazon EC2 instances.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

///<summary>
/// This example shows how to list your Amazon Elastic Compute Cloud
/// (Amazon EC2) instances. It was created using the AWS SDK for .NET
/// version 3.7 and .NET Core 5.0.
///</summary>
public class DescribeInstances
{
    ///<summary>
    /// The Main method creates the Amazon EC2 client object and then calls
    /// first GetInstanceDescriptions and then GetInstanceDescriptionsFiltered
    /// to display the list of Amazon EC2 Instances attached to the default
    /// account.
    ///</summary>
    public static async Task Main()
    {
        // If the Region of the EC2 instances you want to list is different
        // from the default user's Region, you need to specify the Region
        // when you create the client object.
        // For example: RegionEndpoint.USWest1.
        var eC2Client = new AmazonEC2Client();

        // List all EC2 instances.
        await GetInstanceDescriptions(eC2Client);

        string tagName = "IncludeInList";
        string tagValue = "Yes";
        await GetInstanceDescriptionsFiltered(eC2Client, tagName, tagValue);
    }

    ///<summary>
    /// This method uses a paginator to list all of the EC2 Instances
}
```

```
    /// attached to the default account.
    /// </summary>
    /// <param name="client">The Amazon EC2 client object used to call
    /// the DescribeInstances method.</param>
    public static async Task GetInstanceDescriptions(AmazonEC2Client client)
    {
        var request = new DescribeInstancesRequest();

        Console.WriteLine("Showing all instances:");
        var paginator = client.Paginator.DescribeInstances(request);

        await foreach (var response in paginator.Responses)
        {
            foreach (var reservation in response.Reservations)
            {
                foreach (var instance in reservation.Instances)
                {
                    Console.Write($"Instance ID: {instance.InstanceId}");
                    Console.WriteLine($"\\tCurrent State: {instance.State.Name}");
                }
            }
        }
    }

    /// <summary>
    /// This method lists the EC2 instances for this account which have set
    /// the tag named in the tagName parameter with the value in the tagValue
    /// parameter.
    /// </summary>
    /// <param name="client">The Amazon EC2 client object used to call
    /// the DescribeInstances method.</param>
    /// <param name="tagName">A string representing the name of the tag to
    /// filter on.</param>
    /// <param name="tagValue">A string representing the value of the tag
    /// to filter on.</param>
    public static async Task GetInstanceDescriptionsFiltered(AmazonEC2Client
client,
    string tagName, string tagValue)
{
    // This is the tag we want to use to filter
    // the results of our list of instances.
    var filters = new List<Filter>
    {
        new Filter
        {
            Name = $"tag:{tagName}",
            Values = new List<string>
            {
                tagValue,
            },
        };
    };
    var request = new DescribeInstancesRequest
    {
        Filters = filters,
    };

    Console.WriteLine("\nShowing instances with tag: \"IncludeInList\" set to
\"Yes\"");
    var paginator = client.Paginator.DescribeInstances(request);

    await foreach (var response in paginator.Responses)
    {
        foreach (var reservation in response.Reservations)
        {
            foreach (var instance in reservation.Instances)
```

```
        {
            Console.WriteLine($"Instance ID: {instance.InstanceId}");
            Console.WriteLine($"Current State: {instance.State.Name}");
        }
    }
}
```

- For API details, see [DescribeInstances](#) in *AWS SDK for .NET API Reference*.

## Reboot an instance

The following code example shows how to reboot an Amazon EC2 instance.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to reboot Amazon Elastic Compute Cloud (Amazon EC2) instances
/// using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class RebootInstances
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and then calls
    /// RebootInstancesAsync to reboot the instance(s) in the ec2InstanceId
    /// list.
    /// </summary>
    public static async Task Main()
    {
        string ec2InstanceId = "i-0123456789abcdef0";

        // If your EC2 instances are not in the same AWS Region as
        // the default users on your system, you need to supply
        // the AWS Region as a parameter to the client constructor.
        var client = new AmazonEC2Client();

        var request = new RebootInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

        var response = await client.RebootInstancesAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("Instance(s) successfully rebooted.");
        }
        else
        {
            Console.WriteLine("Could not reboot one or more instances.");
        }
    }
}
```

```
    }
```

- For API details, see [RebootInstances in AWS SDK for .NET API Reference](#).

## Start an instance

The following code example shows how to start an Amazon EC2 instance.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

///<summary>
/// Shows how to start a list of Amazon Elastic Compute Cloud (Amazon EC2)
/// instances using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
///</summary>
public class StartInstances
{
    ///<summary>
    /// Initializes the Amazon EC2 client object and uses it to call the
    /// StartInstancesAsync method to start the listed Amazon EC2 instances.
    ///</summary>
    public static async Task Main()
    {
        string ec2InstanceId = "i-0123456789abcdef0";

        // If your EC2 instances are not in the same AWS Region as
        // the default users on your system, you need to supply
        // the AWS Region as a parameter to the client constructor.
        var client = new AmazonEC2Client();

        var request = new StartInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

        var response = await client.StartInstancesAsync(request);

        if (response.StartingInstances.Count > 0)
        {
            var instances = response.StartingInstances;
            instances.ForEach(i =>
            {
                Console.WriteLine($"Successfully started the EC2 Instance with
InstanceID: {i.InstanceId}.");
            });
        }
    }
}
```

- For API details, see [StartInstances in AWS SDK for .NET API Reference](#).

## Stop an instance

The following code example shows how to stop an Amazon EC2 instance.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// Shows how to stop a list of Amazon Elastic Compute Cloud (Amazon EC2)
/// instances using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class StopInstances
{
    /// <summary>
    /// Initializes the Amazon EC2 client object and uses it to call the
    /// StartInstancesAsync method to stop the listed Amazon EC2 instances.
    /// </summary>
    public static async Task Main()
    {
        string ec2InstanceId = "i-0123456789abcdef0";

        // If your EC2 instances are not in the same AWS Region as
        // the default user on your system, you need to supply
        // the AWS Region as a parameter to the client constructor.
        var client = new AmazonEC2Client();

        // In addition to the list of instance Ids, the
        // request can also include the following properties:
        // Force      When true forces the instances to
        //             stop but you have to check the integrity
        //             of the file system. Not recommended on
        //             Windows instances.
        // Hibernate  When true, hibernates the instance if the
        //             instance was enabled for hibernation when
        //             it was launched.
        var request = new StopInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

        var response = await client.StopInstancesAsync(request);

        if (response.StoppingInstances.Count > 0)
        {
            var instances = response.StoppingInstances;
            instances.ForEach(i =>
            {
                Console.WriteLine($"Successfully stopped the EC2 Instance " +
                    $"with InstanceID: {i.InstanceId}.");
            });
        }
    }
}
```

- For API details, see [StopInstances](#) in *AWS SDK for .NET API Reference*.

## Amazon EC2 Auto Scaling examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon EC2 Auto Scaling.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2355\)](#)
- [Scenarios \(p. 2363\)](#)

## Actions

### Create a group

The following code example shows how to create an Auto Scaling group.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Creates a new Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="client">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
/// <param name="groupName">The name to use for the new Auto Scaling
/// group.</param>
/// <param name="launchTemplateName">The name of the Amazon EC2 launch template
/// to use to create instances in the group.</param>
/// <param name="serviceLinkedRoleARN">The AWS Identity and Access
/// Management (IAM) service-linked role that provides the permissions
/// to use with the Auto Scaling group.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the operation.</returns>
public static async Task<bool> CreateAutoScalingGroup(
    AmazonAutoScalingClient client,
    string groupName,
    string launchTemplateName,
    string serviceLinkedRoleARN)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };

    var zoneList = new List<string>
    {
        "us-east-2a",
    };
}
```

```
var request = new CreateAutoScalingGroupRequest
{
    AutoScalingGroupName = groupName,
    AvailabilityZones = zoneList,
    LaunchTemplate = templateSpecification,
    MaxSize = 1,
    MinSize = 1,
    ServiceLinkedRoleARN = serviceLinkedRoleARN,
};

var response = await client.CreateAutoScalingGroupAsync(request);
Console.WriteLine(groupName + " Auto Scaling Group created");
return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [CreateAutoScalingGroup](#) in *AWS SDK for .NET API Reference*.

## Delete a group

The following code example shows how to delete an Auto Scaling group.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Deletes an Auto Scaling group.
/// </summary>
/// <param name="autoScalingClient">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the operation.</returns>
public static async Task<bool> DeleteAutoScalingGroupAsync(
    AmazonAutoScalingClient autoScalingClient,
    string groupName)
{
    var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
    {
        AutoScalingGroupName = groupName,
        ForceDelete = true,
    };

    var response = await
autoScalingClient.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"You successfully deleted {groupName}");
        return true;
    }

    Console.WriteLine($"Couldn't delete {groupName}.");
    return false;
}
```

- For API details, see [DeleteAutoScalingGroup](#) in *AWS SDK for .NET API Reference*.

### Disable metrics collection for a group

The following code example shows how to disable CloudWatch metrics collection for an Auto Scaling group.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Disables the collection of metric data for an Auto Scaling group.
/// </summary>
/// <param name="client">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the operation.</returns>
public static async Task<bool>
DisableMetricsCollectionAsync(AmazonAutoScalingClient client, string groupName)
{
    var request = new DisableMetricsCollectionRequest
    {
        AutoScalingGroupName = groupName,
    };

    var response = await client.DisableMetricsCollectionAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [DisableMetricsCollection](#) in *AWS SDK for .NET API Reference*.

### Enable metrics collection for a group

The following code example shows how to enable CloudWatch metrics collection for an Auto Scaling group.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Enables the collection of metric data for an Auto Scaling group.
/// </summary>
/// <param name="client">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the operation.</returns>
public static async Task<bool>
EnableMetricsCollectionAsync(AmazonAutoScalingClient client, string groupName)
```

```
{  
    var listMetrics = new List<string>  
    {  
        "GroupMaxSize",  
    };  
  
    var collectionRequest = new EnableMetricsCollectionRequest  
    {  
        AutoScalingGroupName = groupName,  
        Metrics = listMetrics,  
        Granularity = "1Minute",  
    };  
  
    var response = await  
client.EnableMetricsCollectionAsync(collectionRequest);  
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;  
}
```

- For API details, see [EnableMetricsCollection](#) in *AWS SDK for .NET API Reference*.

## Get information about groups

The following code example shows how to get information about Auto Scaling groups.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>  
/// Gets data about the instances in an Amazon EC2 Auto Scaling group.  
///</summary>  
///<param name="client">The initialized Amazon EC2 Auto Scaling  
/// client object.</param>  
///<param name="groupName">The name of the Auto Scaling group.</param>  
///<returns>A list of Auto Scaling details.</returns>  
public static async Task<List<AutoScalingInstanceDetails>>  
DescribeAutoScalingInstancesAsync(  
    AmazonAutoScalingClient client,  
    string groupName)  
{  
    var groups = await DescribeAutoScalingGroupsAsync(client, groupName);  
    var instanceIds = new List<string>();  
    groups.ForEach(group =>  
    {  
        if (group.AutoScalingGroupName == groupName)  
        {  
            group.Instances.ForEach(instance =>  
            {  
                instanceIds.Add(instance.InstanceId);  
            });  
        }  
    });  
  
    var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest  
    {  
        MaxRecords = 10,  
        InstanceIds = instanceIds,
```

```
};

    var response = await
client.DescribeAutoScalingInstancesAsync(scalingGroupsRequest);
    var instanceDetails = response.AutoScalingInstances;

    return instanceDetails;
}
```

- For API details, see [DescribeAutoScalingGroups](#) in *AWS SDK for .NET API Reference*.

## Get information about instances

The following code example shows how to get information about Auto Scaling instances.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Gets data about the instances in an Amazon EC2 Auto Scaling group.
///</summary>
///<param name="client">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
///<param name="groupName">The name of the Auto Scaling group.</param>
///<returns>A list of Auto Scaling details.</returns>
public static async Task<List<AutoScalingInstanceDetails>>
DescribeAutoScalingInstancesAsync(
    AmazonAutoScalingClient client,
    string groupName)
{
    var groups = await DescribeAutoScalingGroupsAsync(client, groupName);
    var instanceIds = new List<string>();
    groups.ForEach(group =>
    {
        if (group.AutoScalingGroupName == groupName)
        {
            group.Instances.ForEach(instance =>
            {
                instanceIds.Add(instance.InstanceId);
            });
        }
    });

    var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
    {
        MaxRecords = 10,
        InstanceIds = instanceIds,
    };

    var response = await
client.DescribeAutoScalingInstancesAsync(scalingGroupsRequest);
    var instanceDetails = response.AutoScalingInstances;

    return instanceDetails;
}
```

- For API details, see [DescribeAutoScalingInstances](#) in *AWS SDK for .NET API Reference*.

## Get information about scaling activities

The following code example shows how to get information about Auto Scaling activities.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Retrieves a list of the Auto Scaling activities for an Auto Scaling group.
/// </summary>
/// <param name="client">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A list of Auto Scaling activities.</returns>
public static async Task<List<Activity>> DescribeAutoScalingActivitiesAsync(
    AmazonAutoScalingClient client,
    string groupName)
{
    var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
    {
        AutoScalingGroupName = groupName,
        MaxRecords = 10,
    };

    var response = await
client.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
    return response.Activities;
}
```

- For API details, see [DescribeScalingActivities](#) in *AWS SDK for .NET API Reference*.

## Set the desired capacity of a group

The following code example shows how to set the desired capacity of an Auto Scaling group.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Sets the desired capacity of an Auto Scaling group.
/// </summary>
/// <param name="client">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <param name="desiredCapacity">The desired capacity for the Auto
```

```
///> Scaling group.</param>
///> <returns>A Boolean value that indicates the success or failure of
///> the operation.</returns>
public static async Task<bool> SetDesiredCapacityAsync(
    AmazonAutoScalingClient client,
    string groupName,
    int desiredCapacity)
{
    var capacityRequest = new SetDesiredCapacityRequest
    {
        AutoScalingGroupName = groupName,
        DesiredCapacity = desiredCapacity,
    };

    var response = await client.SetDesiredCapacityAsync(capacityRequest);
    Console.WriteLine($"You have set the DesiredCapacity to
{desiredCapacity}.");

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [SetDesiredCapacity](#) in *AWS SDK for .NET API Reference*.

## Terminate an instance in a group

The following code example shows how to terminate an instance in an Auto Scaling group.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///> <summary>
///> Terminates all instances in the Auto Scaling group in preparation for
///> deleting the group.
///> </summary>
///> <param name="client">The initialized Amazon EC2 Auto Scaling
///> client object.</param>
///> <param name="instanceId">The instance Id of the instance to terminate.</
param>
///> <returns>A Boolean value that indicates the success or failure of
///> the operation.</returns>
public static async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
    AmazonAutoScalingClient client,
    string instanceId)
{
    var request = new TerminateInstanceInAutoScalingGroupRequest
    {
        InstanceId = instanceId,
        ShouldDecrementDesiredCapacity = false,
    };

    var response = await
client.TerminateInstanceInAutoScalingGroupAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"You have terminated the instance {instanceId}");
    }
}
```

```
        return true;
    }

    Console.WriteLine($"Could not terminate {instanceId}");
    return false;
}
```

- For API details, see [TerminateInstanceInAutoScalingGroup](#) in *AWS SDK for .NET API Reference*.

## Update a group

The following code example shows how to update the configuration for an Auto Scaling group.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Updates the capacity of an Auto Scaling group.
/// </summary>
/// <param name="client">The initialized Amazon EC2 Auto Scaling
/// client object.</param>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
/// <param name="serviceLinkedRoleARN">The Amazon Resource Name (ARN)
/// of the AWS Identity and Access Management (IAM) service-linked role.</
param>
/// <param name="maxSize">The maximum number of instances that can be
/// created for the Auto Scaling group.</param>
/// <returns>A Boolean value that indicates the success or failure of
/// the update operation.</returns>
public static async Task<bool> UpdateAutoScalingGroupAsync(
    AmazonAutoScalingClient client,
    string groupName,
    string launchTemplateName,
    string serviceLinkedRoleARN,
    int maxSize)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };

    var groupRequest = new UpdateAutoScalingGroupRequest
    {
        MaxSize = maxSize,
        ServiceLinkedRoleARN = serviceLinkedRoleARN,
        AutoScalingGroupName = groupName,
        LaunchTemplate = templateSpecification,
    };

    var response = await client.UpdateAutoScalingGroupAsync(groupRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"You successfully updated the Auto Scaling group
{groupName}.");
    }
}
```

```
        return true;
    }
    else
    {
        return false;
    }
}
```

- For API details, see [UpdateAutoScalingGroup](#) in *AWS SDK for .NET API Reference*.

## Scenarios

### Manage groups and instances

The following code example shows how to:

- Create an Amazon EC2 Auto Scaling group and configure it with a launch template and Availability Zones.
- Get information about the group and running instances.
- Enable Amazon CloudWatch metrics collection on the group.
- Update the desired capacity of the group and wait for an instance to start.
- Terminate an instance in the group.
- List scaling activities that occur in response to user requests and capacity changes.
- Get statistics for CloudWatch metrics that are collected during the example.
- Stop collecting metrics, terminate all instances, and delete the group.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
global using Amazon;
global using Amazon.AutoScaling;
global using Amazon.AutoScaling.Model;
global using AutoScale_Basics;

var imageId = "ami-05803413c51f242b7";
var instanceType = "t2.micro";
var launchTemplateName = "AutoScaleLaunchTemplate";

// The name of the Auto Scaling group.
var groupName = "AutoScaleExampleGroup";

// The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM)
// service-linked role.
var serviceLinkedRoleARN = "<Enter Value>";

var client = new AmazonAutoScalingClient(RegionEndpoint.USEast2);

Console.WriteLine("Auto Scaling Basics");
DisplayDescription();

// Create the launch template and save the template Id to use when deleting the
```

```
// launch template at the end of the application.  
var launchTemplateId = await EC2Methods.CreateLaunchTemplateAsync(imageId,  
    instanceType, launchTemplateName);  
  
// Confirm that the template was created by asking for a description of it.  
await EC2Methods.DescribeLaunchTemplateAsync(launchTemplateName);  
  
PressEnter();  
  
Console.WriteLine($"--- Creating an Auto Scaling group named {groupName}. ---");  
var success = await AutoScaleMethods.CreateAutoScalingGroup(  
    client,  
    groupName,  
    launchTemplateName,  
    serviceLinkedRoleARN);  
  
// Keep checking the details of the new group until its lifecycle state  
// is "InService".  
Console.WriteLine($"Waiting for the Auto Scaling group to be active.");  
  
List<AutoScalingInstanceDetails> instanceDetails;  
  
do  
{  
    instanceDetails = await AutoScaleMethods.DescribeAutoScalingInstancesAsync(client,  
        groupName);  
}  
while (instanceDetails.Count <= 0);  
  
Console.WriteLine($"Auto scaling group {groupName} successfully created.");  
Console.WriteLine($"{instanceDetails.Count} instances were created for the group.");  
  
// Display the details of the Auto Scaling group.  
instanceDetails.ForEach(detail =>  
{  
    Console.WriteLine($"Group name: {detail.AutoScalingGroupName}");  
});  
  
PressEnter();  
  
Console.WriteLine($"\\n--- Enable metrics collection for {groupName}");  
await AutoScaleMethods.EnableMetricsCollectionAsync(client, groupName);  
  
// Show the metrics that are collected for the group.  
  
// Update the maximum size of the group to three instances.  
Console.WriteLine(" --- Update the Auto Scaling group to increase max size to 3 ---");  
int maxSize = 3;  
await AutoScaleMethods.UpdateAutoScalingGroupAsync(client, groupName,  
    launchTemplateName, serviceLinkedRoleARN, maxSize);  
  
Console.WriteLine(" --- Describe all Auto Scaling groups to show the current state of  
    the group ---");  
var groups = await AutoScaleMethods.DescribeAutoScalingGroupsAsync(client, groupName);  
  
DisplayGroupDetails(groups);  
  
PressEnter();  
  
Console.WriteLine(" --- Describe account limits ---");  
await AutoScaleMethods.DescribeAccountLimitsAsync(client);  
  
Console.WriteLine("Wait 1 min for the resources, including the instance. Otherwise, an  
empty instance Id is returned");  
System.Threading.Thread.Sleep(60000);
```

```
Console.WriteLine("--- Set desired capacity to 2 ---");
int desiredCapacity = 2;
await AutoScaleMethods.SetDesiredCapacityAsync(client, groupName, desiredCapacity);

Console.WriteLine("--- Get the two instance Id values and state ---");

// Empty the group before getting the details again.
groups.Clear();
groups = await AutoScaleMethods.DescribeAutoScalingGroupsAsync(client, groupName);
if (groups is not null)
{
    foreach (AutoScalingGroup group in groups)
    {
        Console.WriteLine($"The group name is {group.AutoScalingGroupName}");
        Console.WriteLine($"The group ARN is {group.AutoScalingGroupARN}");
        var instances = group.Instances;
        foreach (Instance instance in instances)
        {
            Console.WriteLine($"The instance id is {instance.InstanceId}");
            Console.WriteLine($"The lifecycle state is {instance.LifecycleState}");
        }
    }
}

Console.WriteLine("**** List the scaling activities that have occurred for the group");
var activities = await AutoScaleMethods.DescribeAutoScalingActivitiesAsync(client,
    groupName);
if (activities is not null)
{
    activities.ForEach(activity =>
    {
        Console.WriteLine($"The activity Id is {activity.ActivityId}");
        Console.WriteLine($"The activity details are {activity.Details}");
    });
}

// Display the Amazon CloudWatch metrics that have been collected.
var metrics = await CloudWatchMethods.GetCloudWatchMetricsAsync(groupName);
Console.WriteLine($"Metrics collected for {groupName}:");
metrics.ForEach(metric =>
{
    Console.Write($"Metric name: {metric.MetricName}\t");
    Console.WriteLine($"Namespace: {metric.Namespace}");
});

var dataPoints = await CloudWatchMethods.GetMetricStatisticsAsync(groupName);
Console.WriteLine("Details for the metrics collected:");
dataPoints.ForEach(detail =>
{
    Console.WriteLine(detail);
});

// Disable metrics collection.
Console.WriteLine("Disabling the collection of metrics for {groupName}.");
success = await AutoScaleMethods.DisableMetricsCollectionAsync(client, groupName);

if (success)
{
    Console.WriteLine($"Successfully stopped metrics collection for {groupName}.");
}
else
{
    Console.WriteLine($"Could not stop metrics collection for {groupName}.");
}

// Terminate all instances in the group.
```

```
Console.WriteLine(" --- Now terminating all instances in the AWS Auto Scaling group  
 ---");  
  
if (groups is not null)  
{  
    groups.ForEach(group =>  
    {  
        // Only delete instances in the AutoScaling group we created.  
        if (group.AutoScalingGroupName == groupName)  
        {  
            group.Instances.ForEach(async instance =>  
            {  
                await AutoScaleMethods.TerminateInstanceInAutoScalingGroupAsync(client,  
instance.InstanceId);  
            });  
        }  
    });  
}  
  
// After all instances are terminated, delete the group.  
Console.WriteLine(" --- Deleting the Auto Scaling group ---");  
await AutoScaleMethods.DeleteAutoScalingGroupAsync(client, groupName);  
  
// Delete the launch template.  
var deletedLaunchTemplateName = await  
    EC2Methods.DeleteLaunchTemplateAsync(launchTemplateId);  
  
if (deletedLaunchTemplateName == launchTemplateName)  
{  
    Console.WriteLine("Successfully deleted the launch template.");  
}  
  
Console.WriteLine("The demo is now concluded.");  
  
void DisplayDescription()  
{  
    Console.WriteLine("This code example performs the following operations:");  
    Console.WriteLine(" 1. Creates an Amazon EC2 launch template.");  
    Console.WriteLine(" 2. Creates an Auto Scaling group.");  
    Console.WriteLine(" 3. Shows the details of the new Auto Scaling group");  
    Console.WriteLine("      to show that only one instance was created.");  
    Console.WriteLine(" 4. Enables metrics collection.");  
    Console.WriteLine(" 5. Updates the AWS Auto Scaling group to increase the");  
    Console.WriteLine("      capacity to three.");  
    Console.WriteLine(" 6. Describes Auto Scaling groups again to show the");  
    Console.WriteLine("      current state of the group.");  
    Console.WriteLine(" 7. Changes the desired capacity of the Auto Scaling");  
    Console.WriteLine("      group to use an additional instance.");  
    Console.WriteLine(" 8. Shows that there are now instances in the group.");  
    Console.WriteLine(" 9. Lists the scaling activities that have occurred for the  
group.");  
    Console.WriteLine("10. Displays the Amazon CloudWatch metrics that have");  
    Console.WriteLine("      been collected.");  
    Console.WriteLine("11. Disables metrics collection.");  
    Console.WriteLine("12. Terminates all instances in the Auto Scaling group.");  
    Console.WriteLine("13. Deletes the Auto Scaling group.");  
    Console.WriteLine("14. Deletes the Amazon EC2 launch template.");  
}  
  
void DisplayGroupDetails(List<AutoScalingGroup> groups)  
{  
    if (groups is null)  
        return;  
  
    groups.ForEach(group =>  
    {
```

```

        Console.WriteLine($"Group name:\t{group.AutoScalingGroupName}");
        Console.WriteLine($"Group created:\t{group.CreatedTime}");
        Console.WriteLine($"Maximum number of instances:\t{group.MaxSize}");
        Console.WriteLine($"Desired number of instances:\t{group.DesiredCapacity}");
    });
}

void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\n\n");
}

```

Define functions that are called by the scenario to manage launch templates and metrics. These functions wrap Amazon EC2 and CloudWatch actions.

```

using Amazon.EC2;
using Amazon.EC2.Model;

/// <summary>
/// The methods in this class create and delete an Amazon Elastic Compute
/// Cloud (Amazon EC2) launch template for use by the Amazon EC2 Auto
/// Scaling scenario.
/// </summary>
public class EC2Methods
{
    /// <summary>
    /// Create a new Amazon EC2 launch template.
    /// </summary>
    /// <param name="imageId">The image Id to use for instances launched
    /// using the Amazon EC2 launch template.</param>
    /// <param name="instanceType">The type of EC2 instances to create.</param>
    /// <param name="launchTemplateName">The name of the launch template.</param>
    /// <returns>Returns the TemplateID of the new launch template.</returns>
    public static async Task<string> CreateLaunchTemplateAsync(
        string imageId,
        string instanceType,
        string launchTemplateName)
    {
        var client = new AmazonEC2Client();

        var request = new CreateLaunchTemplateRequest
        {
            LaunchTemplateData = new RequestLaunchTemplateData
            {
                ImageId = imageId,
                InstanceType = instanceType,
            },
            LaunchTemplateName = launchTemplateName,
        };

        var response = await client.CreateLaunchTemplateAsync(request);

        return response.LaunchTemplate.LaunchTemplateId;
    }

    /// <summary>
    /// Deletes an Amazon EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateId">The TemplateId of the launch template to
    /// delete.</param>

```

```

    /// <returns>The name of the EC2 launch template that was deleted.</returns>
    public static async Task<string> DeleteLaunchTemplateAsync(string
launchTemplateId)
    {
        var client = new AmazonEC2Client();

        var request = new DeleteLaunchTemplateRequest
        {
            LaunchTemplateId = launchTemplateId,
        };

        var response = await client.DeleteLaunchTemplateAsync(request);
        return response.LaunchTemplate.LaunchTemplateName;
    }

    /// <summary>
    /// Retrieves a information about an EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public static async Task<bool> DescribeLaunchTemplateAsync(string
launchTemplateName)
    {
        var client = new AmazonEC2Client();

        var request = new DescribeLaunchTemplatesRequest
        {
            LaunchTemplateNames = new List<string> { launchTemplateName, },
        };

        var response = await client.DescribeLaunchTemplatesAsync(request);

        if (response.LaunchTemplates != null)
        {
            response.LaunchTemplates.ForEach(template =>
            {
                Console.WriteLine($"{template.LaunchTemplateName}\t");
                Console.WriteLine(template.LaunchTemplateId);
            });
        }

        return true;
    }

    return false;
}

using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

    /// <summary>
    /// The method of this class display the metrics collected for the Amazon
    /// EC2 Auto Scaling group created by the Amazon EC2 Auto Scaling scenario.
    /// </summary>
public class CloudWatchMethods
{
    /// <summary>
    /// Retrieves the metrics information collection for the Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A list of Metrics collected for the Auto Scaling group.</returns>
    public static async Task<List<Metric>> GetCloudWatchMetricsAsync(string
groupName)

```

```

{
    var client = new AmazonCloudWatchClient();

    var filter = new DimensionFilter
    {
        Name = "AutoScalingGroupName",
        Value = $"{groupName}",
    };

    var request = new ListMetricsRequest
    {
        MetricName = "AutoScalingGroupName",
        Dimensions = new List<DimensionFilter> { filter },
        Namespace = "AWS/AutoScaling",
    };

    var response = await client.ListMetricsAsync(request);

    return response.Metrics;
}

/// <summary>
/// Retrieves the metric data collected for an Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</param>
public static async Task<List<Datapoint>> GetMetricStatisticsAsync(string
groupName)
{
    var client = new AmazonCloudWatchClient();

    var metricDimensions = new List<Dimension>
    {
        new Dimension
        {
            Name = "AutoScalingGroupName",
            Value = $"{groupName}",
        },
    };

    // The start time will be yesterday.
    var startTime = DateTime.UtcNow.AddDays(-1);

    var request = new GetMetricStatisticsRequest
    {
        MetricName = "AutoScalingGroupName",
        Dimensions = metricDimensions,
        Namespace = "AWS/AutoScaling",
        Period = 60, // 60 seconds
        Statistics = new List<string>() { "Minimum" },
        StartTimeUtc = startTime,
        EndTimeUtc = DateTime.UtcNow,
    };

    var response = await client.GetMetricStatisticsAsync(request);

    return response.Datapoints;
}

```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [CreateAutoScalingGroup](#)

- [DeleteAutoScalingGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAutoScalingInstances](#)
- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## AWS Glue examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with AWS Glue.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2370\)](#)
- [Scenarios \(p. 2379\)](#)

## Actions

### Create a crawler

The following code example shows how to create an AWS Glue crawler.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Creates an AWS Glue crawler.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="iam">The Amazon Resource Name (ARN) of the IAM role
/// that is used by the crawler.</param>
/// <param name="s3Path">The path to the Amazon S3 bucket where
/// data is stored.</param>
/// <param name="cron">The name of the CRON job that runs the crawler.</param>
/// <param name="dbName">The name of the database.</param>
/// <param name="crawlerName">The name of the AWS Glue crawler.</param>
/// <returns>A Boolean value indicating whether the AWS Glue crawler was
/// created successfully.</returns>
public static async Task<bool> CreateGlueCrawlerAsync(
    AmazonGlueClient glueClient,
    string iam,
    string s3Path,
    string cron,
    string dbName,
```

```
        string crawlerName)
{
    var s3Target = new S3Target
    {
        Path = s3Path,
    };

    var targetList = new List<S3Target>
    {
        s3Target,
    };

    var targets = new CrawlerTargets
    {
        S3Targets = targetList,
    };

    var crawlerRequest = new CreateCrawlerRequest
    {
        DatabaseName = dbName,
        Name = crawlerName,
        Description = "Created by the AWS Glue .NET API",
        Targets = targets,
        Role = iam,
        Schedule = cron,
    };

    var response = await glueClient.CreateCrawlerAsync(crawlerRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"'{crawlerName}' was successfully created");
        return true;
    }
    else
    {
        Console.WriteLine($"Could not create {crawlerName}.");
        return false;
    }
}
```

- For API details, see [CreateCrawler](#) in *AWS SDK for .NET API Reference*.

## Create a job definition

The following code example shows how to create an AWS Glue job definition.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Creates an AWS Glue job.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="jobName">The name of the job to create.</param>
/// <param name="iam">The Amazon Resource Name (ARN) of the IAM role
/// that will be used by the job.</param>
```

```
    /// <param name="scriptLocation">The location where the script is stored.</param>
    /// <returns>A Boolean value indicating whether the AWS Glue job was
    /// created successfully.</returns>
    public static async Task<bool> CreateJobAsync(AmazonGlueClient glueClient,
string jobName, string iam, string scriptLocation)
{
    var command = new JobCommand
    {
        PythonVersion = "3",
        Name = "MyJob1",
        ScriptLocation = scriptLocation,
    };

    var jobRequest = new CreateJobRequest
    {
        Description = "A Job created by using the AWS SDK for .NET",
        GlueVersion = "2.0",
        WorkerType = WorkerType.G1X,
        NumberOfWorkers = 10,
        Name = jobName,
        Role = iam,
        Command = command,
    };

    var response = await glueClient.CreateJobAsync(jobRequest);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{jobName} was successfully created.");
        return true;
    }

    Console.WriteLine($"{jobName} could not be created.");
    return false;
}
```

- For API details, see [CreateJob](#) in [AWS SDK for .NET API Reference](#).

## Delete a crawler

The following code example shows how to delete an AWS Glue crawler.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    /// <summary>
    /// Deletes the named AWS Glue crawler.
    /// </summary>
    /// <param name="glueClient">The initialized AWS Glue client.</param>
    /// <param name="crawlerName">The name of the crawler to delete.</param>
    /// <returns>A Boolean value indicating whether the AWS Glue crawler was
    /// deleted successfully.</returns>
    public static async Task<bool> DeleteSpecificCrawlerAsync(AmazonGlueClient
glueClient, string crawlerName)
{
```

```
var deleteCrawlerRequest = new DeleteCrawlerRequest
{
    Name = crawlerName,
};

var response = await glueClient.DeleteCrawlerAsync(deleteCrawlerRequest);

if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
{
    Console.WriteLine($"{crawlerName} was deleted");
    return true;
}

Console.WriteLine($"Could not create {crawlerName}.");
return false;
}
```

- For API details, see [DeleteCrawler in AWS SDK for .NET API Reference](#).

## Delete a database from the Data Catalog

The following code example shows how to delete a database from the AWS Glue Data Catalog.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Deletes an AWS Glue database.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="databaseName">The name of the database to delete.</param>
/// <returns>A Boolean value indicating whether the AWS Glue database was
/// deleted successfully.</returns>
public static async Task<bool> DeleteDatabaseAsync(AmazonGlueClient glueClient,
string databaseName)
{
    var request = new DeleteDatabaseRequest
    {
        Name = databaseName,
    };

    var response = await glueClient.DeleteDatabaseAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{databaseName} was successfully deleted");
        return true;
    }

    Console.WriteLine($"{databaseName} could not be deleted.");
    return false;
}
```

- For API details, see [DeleteDatabase in AWS SDK for .NET API Reference](#).

## Delete a job definition

The following code example shows how to delete an AWS Glue job definition and all associated runs.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Deletes the named job.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="jobName">The name of the job to delete.</param>
/// <returns>A Boolean value indicating whether the AWS Glue job was
/// deleted successfully.</returns>
public static async Task<bool> DeleteJobAsync(AmazonGlueClient glueClient,
string jobName)
{
    var jobRequest = new DeleteJobRequest
    {
        JobName = jobName,
    };

    var response = await glueClient.DeleteJobAsync(jobRequest);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{jobName} was successfully deleted");
        return true;
    }

    Console.WriteLine($"{jobName} could not be deleted.");
    return false;
}
```

- For API details, see [DeleteJob](#) in *AWS SDK for .NET API Reference*.

## Get a crawler

The following code example shows how to get an AWS Glue crawler.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Retrieves information about a specific AWS Glue crawler.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="crawlerName">The name of the crawler.</param>
/// <returns>A Boolean value indicating whether information about
/// the AWS Glue crawler was retrieved successfully.</returns>
public static async Task<bool> GetSpecificCrawlerAsync(AmazonGlueClient
glueClient, string crawlerName)
```

```
{  
    var crawlerRequest = new GetCrawlerRequest  
    {  
        Name = crawlerName,  
    };  
  
    var response = await glueClient.GetCrawlerAsync(crawlerRequest);  
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)  
    {  
        var databaseName = response.Crawler.DatabaseName;  
        Console.WriteLine($"{crawlerName} has the database {databaseName}");  
        return true;  
    }  
  
    Console.WriteLine($"No information regarding {crawlerName} could be  
found.");  
    return false;  
}
```

- For API details, see [GetCrawler](#) in [AWS SDK for .NET API Reference](#).

## Get a database from the Data Catalog

The following code example shows how to get a database from the AWS Glue Data Catalog.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Gets information about the database created for this Glue  
/// example.  
/// </summary>  
/// <param name="glueClient">The initialized Glue client.</param>  
/// <param name="databaseName">The name of the AWS Glue database.</param>  
/// <returns>A Boolean value indicating whether information about  
/// the AWS Glue database was retrieved successfully.</returns>  
public static async Task<bool> GetSpecificDatabaseAsync(  
    AmazonGlueClient glueClient,  
    string databaseName)  
{  
    var databasesRequest = new GetDatabaseRequest  
    {  
        Name = databaseName,  
    };  
  
    var response = await glueClient.GetDatabaseAsync(databasesRequest);  
  
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)  
    {  
        Console.WriteLine($"The Create Time is  
{response.Database.CreateTime}");  
        return true;  
    }  
  
    Console.WriteLine($"No information about {databaseName}.");  
    return false;  
}
```

- For API details, see [GetDatabase](#) in *AWS SDK for .NET API Reference*.

## Get runs of a job

The following code example shows how to get runs of an AWS Glue job.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Retrieves information about an AWS Glue job.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="jobName">The AWS Glue object for which to retrieve run
/// information.</param>
/// <returns>A Boolean value indicating whether information about
/// the AWS Glue job runs was retrieved successfully.</returns>
public static async Task<bool> GetJobRunsAsync(AmazonGlueClient glueClient,
string jobName)
{
    var runsRequest = new GetJobRunsRequest
    {
        JobName = jobName,
        MaxResults = 20,
    };

    var response = await glueClient.GetJobRunsAsync(runsRequest);
    var jobRuns = response.JobRuns;

    if (jobRuns.Count > 0)
    {
        foreach (JobRun jobRun in jobRuns)
        {
            Console.WriteLine($"Job run state is {jobRun.JobRunState}");
            Console.WriteLine($"Job run Id is {jobRun.Id}");
            Console.WriteLine($"The Glue version is {jobRun.GlueVersion}");
        }

        return true;
    }
    else
    {
        Console.WriteLine("No jobs found.");
        return false;
    }
}
```

- For API details, see [GetJobRuns](#) in *AWS SDK for .NET API Reference*.

## Get tables from a database

The following code example shows how to get tables from a database in the AWS Glue Data Catalog.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Gets the tables used by the database for an AWS Glue crawler.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="dbName">The name of the database.</param>
/// <returns>A Boolean value indicating whether information about
/// the AWS Glue tables was retrieved successfully.</returns>
public static async Task<bool> GetGlueTablesAsync(
    AmazonGlueClient glueClient,
    string dbName)
{
    var tableRequest = new GetTablesRequest
    {
        DatabaseName = dbName,
    };

    // Get the list of AWS Glue databases.
    var response = await glueClient.GetTablesAsync(tableRequest);
    var tables = response.TableList;

    if (tables.Count > 0)
    {
        // Displays the list of table names.
        tables.ForEach(table => { Console.WriteLine($"Table name is: {table.Name}"); });
        return true;
    }
    else
    {
        Console.WriteLine("No tables found.");
        return false;
    }
}
```

- For API details, see [GetTables](#) in [AWS SDK for .NET API Reference](#).

## Start a crawler

The following code example shows how to start an AWS Glue crawler.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Starts the named AWS Glue crawler.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="crawlerName">The name of the crawler to start.</param>
```

```
    /// <returns>A Boolean value indicating whether the AWS Glue crawler  
    /// was started successfully.</returns>  
    public static async Task<bool> StartSpecificCrawlerAsync(AmazonGlueClient  
glueClient, string crawlerName)  
    {  
        var crawlerRequest = new StartCrawlerRequest  
        {  
            Name = crawlerName,  
        };  
  
        var response = await glueClient.StartCrawlerAsync(crawlerRequest);  
  
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)  
        {  
            Console.WriteLine($"'{crawlerName}' was successfully started!");  
            return true;  
        }  
  
        Console.WriteLine($"Could not start AWS Glue crawler, '{crawlerName}'.");  
        return false;  
    }
```

- For API details, see [StartCrawler](#) in [AWS SDK for .NET API Reference](#).

## Start a job run

The following code example shows how to start an AWS Glue job run.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    /// <summary>  
    /// Starts an AWS Glue job.  
    /// </summary>  
    /// <param name="glueClient">The initialized Glue client.</param>  
    /// <param name="jobName">The name of the AWS Glue job to start.</param>  
    /// <returns>A Boolean value indicating whether the AWS Glue job  
    /// was started successfully.</returns>  
    public static async Task<bool> StartJobAsync(AmazonGlueClient glueClient,  
string jobName)  
    {  
        var runRequest = new StartJobRunRequest  
        {  
            WorkerType = WorkerType.G1X,  
            NumberOfWorkers = 10,  
            JobName = jobName,  
        };  
  
        var response = await glueClient.StartJobRunAsync(runRequest);  
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)  
        {  
            Console.WriteLine($"{jobName} successfully started. The job run id is  
{response.JobRunId}.");  
            return true;  
        }  
  
        Console.WriteLine($"Could not start {jobName}.");
```

```
        return false;
    }
```

- For API details, see [StartJobRun](#) in [AWS SDK for .NET API Reference](#).

## Scenarios

### Get started running crawlers and jobs

The following code example shows how to:

- Create and run a crawler that crawls a public Amazon Simple Storage Service (Amazon S3) bucket and generates a metadata database that describes the CSV-formatted data it finds.
- List information about databases and tables in your AWS Glue Data Catalog.
- Create and run a job that extracts CSV data from the source Amazon S3 bucket, transforms it by removing and renaming fields, and loads JSON-formatted output into another Amazon S3 bucket.
- List information about job runs and view some of the transformed data.
- Delete all resources created by the demo.

For more information, see [Tutorial: Getting started with AWS Glue Studio](#).

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps AWS Glue functions that are used in the scenario.

```
namespace Glue_Basics
{
    /// <summary>
    /// Methods for working the AWS Glue by using the AWS SDK for .NET (v3.7).
    /// </summary>
    public static class GlueMethods
    {

        /// <summary>
        /// Creates a database for use by an AWS Glue crawler.
        /// </summary>
        /// <param name="glueClient">The initialized AWS Glue client.</param>
        /// <param name="dbName">The name of the new database.</param>
        /// <param name="locationUri">The location of scripts that will be
        /// used by the AWS Glue crawler.</param>
        /// <returns>A Boolean value indicating whether the AWS Glue database
        /// was created successfully.</returns>
        public static async Task<bool> CreateDatabaseAsync(AmazonGlueClient glueClient,
string dbName, string locationUri)
        {
            try
            {
                var DataBaseInput = new DatabaseInput
                {
                    Description = "Built with the AWS SDK for .NET (v3)",
                    Name = dbName,
                    LocationUri = locationUri,
                };
            }
        }
    }
}
```

```
        var request = new CreateDatabaseRequest
        {
            DatabaseInput = DataBaseInput,
        };

        var response = await glueClient.CreateDatabaseAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("The database was successfully created");
            return true;
        }
        else
        {
            Console.WriteLine("Could not create the database.");
            return false;
        }
    }
    catch (AmazonGlueException ex)
    {
        Console.WriteLine($"Error occurred: '{ex.Message}'");
        return false;
    }
}

///<summary>
/// Creates an AWS Glue crawler.
///</summary>
///<param name="glueClient">The initialized AWS Glue client.</param>
///<param name="iam">The Amazon Resource Name (ARN) of the IAM role
/// that is used by the crawler.</param>
///<param name="s3Path">The path to the Amazon S3 bucket where
/// data is stored.</param>
///<param name="cron">The name of the CRON job that runs the crawler.</param>
///<param name="dbName">The name of the database.</param>
///<param name="crawlerName">The name of the AWS Glue crawler.</param>
///<returns>A Boolean value indicating whether the AWS Glue crawler was
/// created successfully.</returns>
public static async Task<bool> CreateGlueCrawlerAsync(
    AmazonGlueClient glueClient,
    string iam,
    string s3Path,
    string cron,
    string dbName,
    string crawlerName)
{
    var s3Target = new S3Target
    {
        Path = s3Path,
    };

    var targetList = new List<S3Target>
    {
        s3Target,
    };

    var targets = new CrawlerTargets
    {
        S3Targets = targetList,
    };

    var crawlerRequest = new CreateCrawlerRequest
    {
        DatabaseName = dbName,
        Name = crawlerName,
```

```
        Description = "Created by the AWS Glue .NET API",
        Targets = targets,
        Role = iam,
        Schedule = cron,
    };

    var response = await glueClient.CreateCrawlerAsync(crawlerRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{crawlerName} was successfully created");
        return true;
    }
    else
    {
        Console.WriteLine($"Could not create {crawlerName}.");
        return false;
    }
}

/// <summary>
/// Creates an AWS Glue job.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="jobName">The name of the job to create.</param>
/// <param name="iam">The Amazon Resource Name (ARN) of the IAM role
/// that will be used by the job.</param>
/// <param name="scriptLocation">The location where the script is stored.</param>
public static async Task<bool> CreateJobAsync(AmazonGlueClient glueClient,
string jobName, string iam, string scriptLocation)
{
    var command = new JobCommand
    {
        PythonVersion = "3",
        Name = "MyJob1",
        ScriptLocation = scriptLocation,
    };

    var jobRequest = new CreateJobRequest
    {
        Description = "A Job created by using the AWS SDK for .NET",
        GlueVersion = "2.0",
        WorkerType = WorkerType.G1X,
        NumberOfWorkers = 10,
        Name = jobName,
        Role = iam,
        Command = command,
    };

    var response = await glueClient.CreateJobAsync(jobRequest);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{jobName} was successfully created.");
        return true;
    }

    Console.WriteLine($"{jobName} could not be created.");
    return false;
}
```

```
    /// <summary>
    /// Deletes the named AWS Glue crawler.
    /// </summary>
    /// <param name="glueClient">The initialized AWS Glue client.</param>
    /// <param name="crawlerName">The name of the crawler to delete.</param>
    /// <returns>A Boolean value indicating whether the AWS Glue crawler was
    /// deleted successfully.</returns>
    public static async Task<bool> DeleteSpecificCrawlerAsync(AmazonGlueClient
glueClient, string crawlerName)
{
    var deleteCrawlerRequest = new DeleteCrawlerRequest
    {
        Name = crawlerName,
    };

    var response = await glueClient.DeleteCrawlerAsync(deleteCrawlerRequest);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{crawlerName} was deleted");
        return true;
    }

    Console.WriteLine($"Could not create {crawlerName}.");
    return false;
}

    /// <summary>
    /// Deletes an AWS Glue database.
    /// </summary>
    /// <param name="glueClient">The initialized AWS Glue client.</param>
    /// <param name="databaseName">The name of the database to delete.</param>
    /// <returns>A Boolean value indicating whether the AWS Glue database was
    /// deleted successfully.</returns>
    public static async Task<bool> DeleteDatabaseAsync(AmazonGlueClient glueClient,
string databaseName)
{
    var request = new DeleteDatabaseRequest
    {
        Name = databaseName,
    };

    var response = await glueClient.DeleteDatabaseAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{databaseName} was successfully deleted");
        return true;
    }

    Console.WriteLine($"{databaseName} could not be deleted.");
    return false;
}

    /// <summary>
    /// Deletes the named job.
    /// </summary>
    /// <param name="glueClient">The initialized AWS Glue client.</param>
    /// <param name="jobName">The name of the job to delete.</param>
    /// <returns>A Boolean value indicating whether the AWS Glue job was
    /// deleted successfully.</returns>
```

```
public static async Task<bool> DeleteJobAsync(AmazonGlueClient glueClient,
string jobName)
{
    var jobRequest = new DeleteJobRequest
    {
        JobName = jobName,
    };

    var response = await glueClient.DeleteJobAsync(jobRequest);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"'{jobName}' was successfully deleted");
        return true;
    }

    Console.WriteLine($"'{jobName}' could not be deleted.");
    return false;
}

/// <summary>
/// Gets a list of AWS Glue jobs.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <returns>A Boolean value indicating whether information about the
/// AWS Glue jobs was retrieved successfully.</returns>
/// <returns>A Boolean value indicating whether information about
/// all AWS Glue jobs was retrieved.</returns>
public static async Task<bool> GetAllJobsAsync(AmazonGlueClient glueClient)
{
    var jobsRequest = new GetJobsRequest
    {
        MaxResults = 10,
    };

    var response = await glueClient.GetJobsAsync(jobsRequest);
    var jobs = response.Jobs;
    if (jobs.Count > 0)
    {
        jobs.ForEach(job => { Console.WriteLine($"The job name is:
{job.Name}"); });
        return true;
    }
    else
    {
        Console.WriteLine("Didn't find any jobs.");
        return false;
    }
}

/// <summary>
/// Gets the tables used by the database for an AWS Glue crawler.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="dbName">The name of the database.</param>
/// <returns>A Boolean value indicating whether information about
/// the AWS Glue tables was retrieved successfully.</returns>
public static async Task<bool> GetGlueTablesAsync(
    AmazonGlueClient glueClient,
    string dbName)
{
    var tableRequest = new GetTablesRequest
```

```
{  
    DatabaseName = dbName,  
};  
  
// Get the list of AWS Glue databases.  
var response = await glueClient.GetTablesAsync(tableRequest);  
var tables = response.TableList;  
  
if (tables.Count > 0)  
{  
    // Displays the list of table names.  
    tables.ForEach(table => { Console.WriteLine($"Table name is:  
{table.Name}"); });  
    return true;  
}  
else  
{  
    Console.WriteLine("No tables found.");  
    return false;  
}  
}  
  
  
/// <summary>  
/// Retrieves information about an AWS Glue job.  
/// </summary>  
/// <param name="glueClient">The initialized AWS Glue client.</param>  
/// <param name="jobName">The AWS Glue object for which to retrieve run  
/// information.</param>  
/// <returns>A Boolean value indicating whether information about  
/// the AWS Glue job runs was retrieved successfully.</returns>  
public static async Task<bool> GetJobRunsAsync(AmazonGlueClient glueClient,  
string jobName)  
{  
    var runsRequest = new GetJobRunsRequest  
    {  
        JobName = jobName,  
        MaxResults = 20,  
    };  
  
    var response = await glueClient.GetJobRunsAsync(runsRequest);  
    var jobRuns = response.JobRuns;  
  
    if (jobRuns.Count > 0)  
    {  
        foreach (JobRun jobRun in jobRuns)  
        {  
            Console.WriteLine($"Job run state is {jobRun.JobRunState}");  
            Console.WriteLine($"Job run Id is {jobRun.Id}");  
            Console.WriteLine($"The Glue version is {jobRun.GlueVersion}");  
        }  
  
        return true;  
    }  
    else  
    {  
        Console.WriteLine("No jobs found.");  
        return false;  
    }  
}  
  
  
/// <summary>  
/// Retrieves information about a specific AWS Glue crawler.
```

```
    /// </summary>
    /// <param name="glueClient">The initialized AWS Glue client.</param>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating whether information about
    /// the AWS Glue crawler was retrieved successfully.</returns>
    public static async Task<bool> GetSpecificCrawlerAsync(AmazonGlueClient
glueClient, string crawlerName)
    {
        var crawlerRequest = new GetCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await glueClient.GetCrawlerAsync(crawlerRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var databaseName = response.Crawler.DatabaseName;
            Console.WriteLine($"'{crawlerName}' has the database {databaseName}");
            return true;
        }

        Console.WriteLine($"No information regarding {crawlerName} could be
found.");
        return false;
    }

    /// <summary>
    /// Gets information about the database created for this Glue
    /// example.
    /// </summary>
    /// <param name="glueClient">The initialized Glue client.</param>
    /// <param name="databaseName">The name of the AWS Glue database.</param>
    /// <returns>A Boolean value indicating whether information about
    /// the AWS Glue database was retrieved successfully.</returns>
    public static async Task<bool> GetSpecificDatabaseAsync(
        AmazonGlueClient glueClient,
        string databaseName)
    {
        var databasesRequest = new GetDatabaseRequest
        {
            Name = databaseName,
        };

        var response = await glueClient.GetDatabaseAsync(databasesRequest);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"The Create Time is
{response.Database.CreateTime}");
            return true;
        }

        Console.WriteLine($"No information about {databaseName}.");
        return false;
    }

    /// <summary>
    /// Starts an AWS Glue job.
    /// </summary>
    /// <param name="glueClient">The initialized Glue client.</param>
    /// <param name="jobName">The name of the AWS Glue job to start.</param>
    /// <returns>A Boolean value indicating whether the AWS Glue job
```

```

    /// was started successfully.</returns>
    public static async Task<bool> StartJobAsync(AmazonGlueClient glueClient,
string jobName)
{
    var runRequest = new StartJobRunRequest
    {
        WorkerType = WorkerType.G1X,
        NumberOfWorkers = 10,
        JobName = jobName,
    };

    var response = await glueClient.StartJobRunAsync(runRequest);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"'{jobName}' successfully started. The job run id is
{response.JobRunId}.");
        return true;
    }

    Console.WriteLine($"Could not start {jobName}.");
    return false;
}

/// <summary>
/// Starts the named AWS Glue crawler.
/// </summary>
/// <param name="glueClient">The initialized AWS Glue client.</param>
/// <param name="crawlerName">The name of the crawler to start.</param>
/// <returns>A Boolean value indicating whether the AWS Glue crawler
/// was started successfully.</returns>
public static async Task<bool> StartSpecificCrawlerAsync(AmazonGlueClient
glueClient, string crawlerName)
{
    var crawlerRequest = new StartCrawlerRequest
    {
        Name = crawlerName,
    };

    var response = await glueClient.StartCrawlerAsync(crawlerRequest);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"{crawlerName} was successfully started!");
        return true;
    }

    Console.WriteLine($"Could not start AWS Glue crawler, {crawlerName}.");
    return false;
}
}

```

Create a class that runs the scenario.

```

global using Amazon.Glue;
global using Amazon.Glue.Model;
global using Glue_Basics;

// This example uses .NET Core 6 and the AWS SDK for .NET (v3.7)

```

```
// Before running the code, set up your development environment,
// including your credentials. For more information, see the
// following topic:
//   https://docs.aws.amazon.com/sdk-for-net/v3/developer-guide/net-dg-config.html
//
// To set up the resources you need, see the following topic:
//   https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
//
// This example performs the following tasks:
//   1. CreateDatabase
//   2. CreateCrawler
//   3. GetCrawler
//   4. StartCrawler
//   5. GetDatabase
//   6. GetTables
//   7. CreateJob
//   8. StartJobRun
//   9. ListJobs
// 10. GetJobRuns
// 11. DeleteJob
// 12. DeleteDatabase
// 13. DeleteCrawler

// Initialize the values that we need for the scenario.
// The Amazon Resource Name (ARN) of the service role used by the crawler.
var iam = "arn:aws:iam::012345678901:role/AWSGlueServiceRole-CrawlerTutorial";

// The path to the Amazon S3 bucket where the comma-delimited file is stored.
var s3Path = "s3://crawler-public-us-east-1/flight/2016/csv";

var cron = "cron(15 12 * * ? *)";

// The name of the database used by the crawler.
var dbName = "example-flights-db";

var crawlerName = "Flight Data Crawler";
var jobName = "glue-job34";
var scriptLocation = "s3://aws-glue-scripts-012345678901-us-west-1/GlueDemoUser";
var locationUri = "s3://crawler-public-us-east-1/flight/2016/csv/";

var glueClient = new AmazonGlueClient();
await GlueMethods.DeleteDatabaseAsync(glueClient, dbName);

Console.WriteLine("Creating the database and crawler for the AWS Glue example.");
var success = await GlueMethods.CreateDatabaseAsync(glueClient, dbName, locationUri);
success = await GlueMethods.CreateGlueCrawlerAsync(glueClient, iam, s3Path, cron,
    dbName, crawlerName);

// Get information about the AWS Glue crawler.
Console.WriteLine("Showing information about the newly created AWS Glue crawler.");
success = await GlueMethods.GetSpecificCrawlerAsync(glueClient, crawlerName);

Console.WriteLine("Starting the new AWS Glue crawler.");
success = await GlueMethods.StartSpecificCrawlerAsync(glueClient, crawlerName);

Console.WriteLine("Displaying information about the database used by the crawler.");
success = await GlueMethods.GetSpecificDatabaseAsync(glueClient, dbName);
success = await GlueMethods.GetGlueTablesAsync(glueClient, dbName);

Console.WriteLine("Creating a new AWS Glue job.");
success = await GlueMethods.CreateJobAsync(glueClient, jobName, iam, scriptLocation);

Console.WriteLine("Starting the new AWS Glue job.");
success = await GlueMethods.StartJobAsync(glueClient, jobName);

Console.WriteLine("Getting information about the AWS Glue job.");
```

```
success = await GlueMethods.GetAllJobsAsync(glueClient);
success = await GlueMethods.GetJobRunsAsync(glueClient, jobName);

Console.WriteLine("Deleting the AWS Glue job used by the example.");
success = await GlueMethods.DeleteJobAsync(glueClient, jobName);

Console.WriteLine("\n*** Waiting 5 MIN for the " + crawlerName + " to stop. ***");
System.Threading.Thread.Sleep(300000);

Console.WriteLine("Clean up the resources created for the example.");
success = await GlueMethods.DeleteDatabaseAsync(glueClient, dbName);
success = await GlueMethods.DeleteSpecificCrawlerAsync(glueClient, crawlerName);

Console.WriteLine("Successfully completed the AWS Glue Scenario ");
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.

- [CreateCrawler](#)
- [CreateJob](#)
- [DeleteCrawler](#)
- [DeleteDatabase](#)
- [DeleteJob](#)
- [DeleteTable](#)
- [GetCrawler](#)
- [GetDatabase](#)
- [GetDatabases](#)
- [GetJob](#)
- [GetJobRun](#)
- [GetJobRuns](#)
- [GetTables](#)
- [ListJobs](#)
- [StartCrawler](#)
- [StartJobRun](#)

## IAM examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with AWS Identity and Access Management.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2388\)](#)
- [Scenarios \(p. 2402\)](#)

## Actions

### Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Attach the policy to the role so that the user can assume it.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="policyArn">The ARN of the policy to attach.</param>
/// <param name="roleName">The name of the role to attach the policy to.</param>
public static async Task AttachRoleAsync(
    AmazonIdentityManagementServiceClient client,
    string policyArn,
    string roleName)
{
    var request = new AttachRolePolicyRequest
    {
        PolicyArn = policyArn,
        RoleName = roleName,
    };

    var response = await client.AttachRolePolicyAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine("Successfully attached the policy to the role.");
    }
    else
    {
        Console.WriteLine("Could not attach the policy.");
    }
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for .NET API Reference*.

## Create a policy

The following code example shows how to create an IAM policy.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create a policy to allow a user to list the buckets in an account.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="policyName">The name of the policy to create.</param>
/// <param name="policyDocument">The permissions policy document.</param>
/// <returns>The newly created ManagedPolicy object.</returns>
public static async Task<ManagedPolicy> CreatePolicyAsync(
    AmazonIdentityManagementServiceClient client,
```

```
        string policyName,
        string policyDocument)
{
    var request = new CreatePolicyRequest
    {
        PolicyName = policyName,
        PolicyDocument = policyDocument,
    };

    var response = await client.CreatePolicyAsync(request);

    return response.Policy;
}
```

- For API details, see [CreatePolicy](#) in *AWS SDK for .NET API Reference*.

## Create a role

The following code example shows how to create an IAM role.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create a new IAM role which we can attach to a user.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="roleName">The name of the IAM role to create.</param>
/// <param name="rolePermissions">The permissions which the role will have.</param>
/// <returns>A Role object representing the newly created role.</returns>
public static async Task<Role> CreateRoleAsync(
    AmazonIdentityManagementServiceClient client,
    string roleName,
    string rolePermissions)
{
    var request = new CreateRoleRequest
    {
        RoleName = roleName,
        AssumeRolePolicyDocument = rolePermissions,
    };

    var response = await client.CreateRoleAsync(request);

    return response.Role;
}
```

- For API details, see [CreateRole](#) in *AWS SDK for .NET API Reference*.

## Create a user

The following code example shows how to create an IAM user.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create a new IAM user.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="userName">A string representing the user name of the
/// new user.</param>
/// <returns>The newly created user.</returns>
public static async Task<User> CreateUserAsync(
    AmazonIdentityManagementServiceClient client,
    string userName)
{
    var request = new CreateUserRequest
    {
        UserName = userName,
    };

    var response = await client.CreateUserAsync(request);

    return response.User;
}
```

- For API details, see [CreateUser](#) in *AWS SDK for .NET API Reference*.

## Create an access key

The following code example shows how to create an IAM access key.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create a new AccessKey for the user.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="userName">The name of the user for whom to create the key.</param>
/// <returns>A new IAM access key for the user.</returns>
public static async Task<AccessKey> CreateAccessKeyAsync(
    AmazonIdentityManagementServiceClient client,
    string userName)
{
    var request = new CreateAccessKeyRequest
    {
        UserName = userName,
    };

    var response = await client.CreateAccessKeyAsync(request);
```

```
        if (response.AccessKey is not null)
    {
        Console.WriteLine($"Successfully created Access Key for {userName}.");
    }

    return response.AccessKey;
}
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for .NET API Reference*.

## Delete a role policy

The following code example shows how to delete an IAM role policy.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

public class DeleteRolePolicy
{
    ///<summary>
    /// Initializes the IAM client object and then calls DeleteRolePolicyAsync
    /// to delete the Policy attached to the Role.
    ///</summary>
    public static async Task Main()
    {
        var client = new AmazonIdentityManagementServiceClient();
        var response = await client.DeleteRolePolicyAsync(new
DeleteRolePolicyRequest
        {
            PolicyName = "ExamplePolicy",
            RoleName = "Test-Role",
        });

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("Policy successfully deleted.");
        }
        else
        {
            Console.WriteLine("Could not delete policy.");
        }
    }
}
```

- For API details, see [DeleteRolePolicy](#) in *AWS SDK for .NET API Reference*.

## Delete a user

The following code example shows how to delete an IAM user.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Delete the user, and other resources created for this example.
/// </summary>
/// <param name="client">The initialized client object.</param>
/// <param name="accessKeyId">The Id of the user's access key.</param>
/// <param name="userName">The user name of the user to delete.</param>
/// <param name="policyName">The name of the policy to delete.</param>
/// <param name="policyArn">The Amazon Resource Name ARN of the Policy to
delete.</param>
/// <param name="roleName">The name of the role that will be deleted.</param>
public static async Task DeleteResourcesAsync(
    AmazonIdentityManagementServiceClient client,
    string accessKeyId,
    string userName,
    string policyArn,
    string roleName)
{
    var detachPolicyResponse = await client.DetachRolePolicyAsync(new
DetachRolePolicyRequest
{
    PolicyArn = policyArn,
    RoleName = roleName,
});

    var delPolicyResponse = await client.DeletePolicyAsync(new
DeletePolicyRequest
{
    PolicyArn = policyArn,
});

    var delRoleResponse = await client.DeleteRoleAsync(new DeleteRoleRequest
{
    RoleName = roleName,
});

    var delAccessKey = await client.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
{
    AccessKeyId = accessKeyId,
    UserName = userName,
});

    var delUserResponse = await client.DeleteUserAsync(new DeleteUserRequest
{
    UserName = userName,
});
}
```

- For API details, see [DeleteUser](#) in *AWS SDK for .NET API Reference*.

## Delete an access key

The following code example shows how to delete an IAM access key.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Delete the user, and other resources created for this example.
/// </summary>
/// <param name="client">The initialized client object.</param>
/// <param name="accessKeyId">The Id of the user's access key.</param>
/// <param name="userName">The user name of the user to delete.</param>
/// <param name="policyName">The name of the policy to delete.</param>
/// <param name="policyArn">The Amazon Resource Name ARN of the Policy to
delete.</param>
/// <param name="roleName">The name of the role that will be deleted.</param>
public static async Task DeleteResourcesAsync(
    AmazonIdentityManagementServiceClient client,
    string accessKeyId,
    string userName,
    string policyArn,
    string roleName)
{
    var detachPolicyResponse = await client.DetachRolePolicyAsync(new
DetachRolePolicyRequest
{
    PolicyArn = policyArn,
    RoleName = roleName,
});

    var delPolicyResponse = await client.DeletePolicyAsync(new
DeletePolicyRequest
{
    PolicyArn = policyArn,
});

    var delRoleResponse = await client.DeleteRoleAsync(new DeleteRoleRequest
{
    RoleName = roleName,
});

    var delAccessKey = await client.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
{
    AccessKeyId = accessKeyId,
    UserName = userName,
});

    var delUserResponse = await client.DeleteUserAsync(new DeleteUserRequest
{
    UserName = userName,
});
}
```

- For API details, see [DeleteAccessKey](#) in [AWS SDK for .NET API Reference](#).

## Detach a policy from a role

The following code example shows how to detach an IAM policy from a role.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Delete the user, and other resources created for this example.
/// </summary>
/// <param name="client">The initialized client object.</param>
/// <param name="accessKeyId">The Id of the user's access key.</param>
/// <param name="userName">The user name of the user to delete.</param>
/// <param name="policyName">The name of the policy to delete.</param>
/// <param name="policyArn">The Amazon Resource Name ARN of the Policy to
delete.</param>
/// <param name="roleName">The name of the role that will be deleted.</param>
public static async Task DeleteResourcesAsync(
    AmazonIdentityManagementServiceClient client,
    string accessKeyId,
    string userName,
    string policyArn,
    string roleName)
{
    var detachPolicyResponse = await client.DetachRolePolicyAsync(new
DetachRolePolicyRequest
{
    PolicyArn = policyArn,
    RoleName = roleName,
});

    var delPolicyResponse = await client.DeletePolicyAsync(new
DeletePolicyRequest
{
    PolicyArn = policyArn,
});

    var delRoleResponse = await client.DeleteRoleAsync(new DeleteRoleRequest
{
    RoleName = roleName,
});

    var delAccessKey = await client.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
{
    AccessKeyId = accessKeyId,
    UserName = userName,
});

    var delUserResponse = await client.DeleteUserAsync(new DeleteUserRequest
{
    UserName = userName,
});
}
```

- For API details, see [DetachRolePolicy](#) in *AWS SDK for .NET API Reference*.

## Get a policy

The following code example shows how to get an IAM policy.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();
var request = new GetPolicyRequest
{
    PolicyArn = "POLICY_ARN",
};

var response = await client.GetPolicyAsync(request);

Console.WriteLine($"{{response.Policy.PolicyName}} was created on ");
Console.WriteLine($"{{response.Policy.CreateDate}}");
```

- For API details, see [GetPolicy](#) in *AWS SDK for .NET API Reference*.

## Get a role

The following code example shows how to get an IAM role.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

var response = await client.GetRoleAsync(new GetRoleRequest
{
    RoleName = "LambdaS3Role",
});

if (response.Role != null)
{
    Console.WriteLine($"{{response.Role.RoleName}} with ARN: {{response.Role.Arn}}");
    Console.WriteLine($"{{response.Role.Description}}");
    Console.WriteLine($"Created: {{response.Role.CreateDate}} Last used on:
{{response.Role.RoleLastUsed}}");
}
```

- For API details, see [GetRole](#) in *AWS SDK for .NET API Reference*.

## Get the account password policy

The following code example shows how to get the IAM account password policy.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

try
{
    var request = new GetAccountPasswordPolicyRequest();
    var response = await client.GetAccountPasswordPolicyAsync(request);

    Console.WriteLine($"{response.PasswordPolicy}");
}
catch (NoSuchEntityException ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}
```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for .NET API Reference*.

## List SAML providers

The following code example shows how to list SAML providers for IAM.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

var response = await client.ListSAMLProvidersAsync(new ListSAMLProvidersRequest());

response.SAMLProviderList.ForEach(samlProvider =>
{
    Console.WriteLine($"{samlProvider.Arn} created on: {samlProvider.CreateDate}");
});
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for .NET API Reference*.

## List groups

The following code example shows how to list IAM groups.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

var request = new ListGroupsRequest
{
    MaxItems = 10,
};

var response = await client.ListGroupsAsync(request);

do
{
    response.Groups.ForEach(group =>
    {
        Console.WriteLine($"{group.GroupName} created on: {group.CreateDate}");
    });

    if (response.IsTruncated)
    {
        request.Marker = response.Marker;
        response = await client.ListGroupsAsync(request);
    }
} while (response.IsTruncated);
```

- For API details, see [ListGroups](#) in *AWS SDK for .NET API Reference*.

## List inline policies for a role

The following code example shows how to list inline policies for an IAM role.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;
using System;

var client = new AmazonIdentityManagementServiceClient();
var request = new ListRolePoliciesRequest
{
    RoleName = "LambdaS3Role",
};
```

```
var response = new ListRolePoliciesResponse();

do
{
    response = await client.ListRolePoliciesAsync(request);

    if (response.PolicyNames.Count > 0)
    {
        response.PolicyNames.ForEach(policyName =>
        {
            Console.WriteLine($"{policyName}");
        });
    }

    // As long as response.IsTruncated is true, set request.Marker equal
    // to response.Marker and call ListRolesAsync again.
    if (response.IsTruncated)
    {
        request.Marker = response.Marker;
    }
} while (response.IsTruncated);
```

- For API details, see [ListRolePolicies in AWS SDK for .NET API Reference](#).

## List policies

The following code example shows how to list IAM policies.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;
using System;

var client = new AmazonIdentityManagementServiceClient();

var request = new ListPoliciesRequest
{
    MaxItems = 10,
};

var response = new ListPoliciesResponse();

do
{
    response = await client.ListPoliciesAsync(request);
    response.Policies.ForEach(policy =>
    {
        Console.Write($"{policy.PolicyName} ");
        Console.Write($"with ID: {policy.PolicyId} ");
        Console.Write($"and ARN: {policy.Arn}. ");
        Console.WriteLine($"It was created on {policy.CreateDate}.");
    });
}

if (response.IsTruncated)
```

```
{  
    request.Marker = response.Marker;  
}  
} while (response.IsTruncated);
```

- For API details, see [ListPolicies](#) in *AWS SDK for .NET API Reference*.

### List policies attached to a role

The following code example shows how to list policies attached to an IAM role.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using Amazon.IdentityManagement;  
using Amazon.IdentityManagement.Model;  
  
var client = new AmazonIdentityManagementServiceClient();  
var request = new ListAttachedRolePoliciesRequest  
{  
    MaxItems = 10,  
    RoleName = "testAssumeRole",  
};  
  
var response = await client.ListAttachedRolePoliciesAsync(request);  
  
do  
{  
    response.AttachedPolicies.ForEach(policy =>  
    {  
        Console.WriteLine($"{policy.PolicyName} with ARN: {policy.PolicyArn}");  
    });  
  
    if (response.IsTruncated)  
    {  
        request.Marker = response.Marker;  
        response = await client.ListAttachedRolePoliciesAsync(request);  
    }  
}  
} while (response.IsTruncated);
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for .NET API Reference*.

### List roles

The following code example shows how to list IAM roles.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

// Without the MaxItems value, the ListRolesAsync method will
// return information for up to 100 roles. If there are more
// than the MaxItems value or more than 100 roles, the response
// value IsTruncated will be true.
var request = new ListRolesRequest
{
    MaxItems = 10,
};

var response = new ListRolesResponse();

do
{
    response = await client.ListRolesAsync(request);
    response.Roles.ForEach(role =>
    {
        Console.WriteLine($"{role.RoleName} - ARN {role.Arn}");
    });

    // As long as response.IsTruncated is true, set request.Marker equal
    // to response.Marker and call ListRolesAsync again.
    if (response.IsTruncated)
    {
        request.Marker = response.Marker;
    }
} while (response.IsTruncated);
```

- For API details, see [ListRoles](#) in [AWS SDK for .NET API Reference](#).

## List users

The following code example shows how to list IAM users.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();
var request = new ListUsersRequest
{
    MaxItems = 10,
};
var response = await client.ListUsersAsync(request);

do
```

```
{  
    response.Users.ForEach(user =>  
    {  
        Console.WriteLine($"'{user.UserName}' created on {user.CreateDate}.");  
        Console.WriteLine($"ARN: {user.Arn}\n");  
    });  
  
    request.Marker = response.Marker;  
    response = await client.ListUsersAsync(request);  
} while (response.IsTruncated);
```

- For API details, see [ListUsers](#) in *AWS SDK for .NET API Reference*.

## Scenarios

### Create a user and assume a role

The following code example shows how to:

- Create a user who has no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list Amazon S3 buckets using temporary credentials.
- Delete the policy, role, and user.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.IO;  
using System.Threading.Tasks;  
using Amazon;  
using Amazon.IdentityManagement;  
using Amazon.IdentityManagement.Model;  
using Amazon.S3;  
using Amazon.SecurityToken;  
using Amazon.SecurityToken.Model;  
  
public class IAM_Basics  
{  
    // Values needed for user, role, and policies.  
    private const string UserName = "example-user";  
    private const string S3PolicyName = "s3-list-buckets-policy";  
    private const string RoleName = "temporary-role";  
    private const string AssumePolicyName = "sts-trust-user";  
  
    private static readonly RegionEndpoint Region = RegionEndpoint.USEast2;  
  
    public static async Task Main()  
    {  
        DisplayInstructions();  
  
        // Create the IAM client object.
```

```
var client = new AmazonIdentityManagementServiceClient(Region);

// First create a user. By default, the new user has
// no permissions.
Console.WriteLine($"Creating a new user with user name: {UserName}.");
var user = await CreateUserAsync(client, UserName);
var userArn = user.Arn;
Console.WriteLine($"Successfully created user: {UserName} with ARN:
{userArn}.");

// Create an AccessKey for the user.
var accessKey = await CreateAccessKeyAsync(client, UserName);

var accessKeyId = accessKey.AccessKeyId;
var secretAccessKey = accessKey.SecretAccessKey;

// Try listing the Amazon Simple Storage Service (Amazon S3)
// buckets. This should fail at this point because the user doesn't
// have permissions to perform this task.
var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
await ListMyBucketsAsync(s3Client1);

// Define a role policy document that allows the new user
// to assume the role.
// string assumeRolePolicyDocument = File.ReadAllText("assumePolicy.json");
string assumeRolePolicyDocument = "{" +
    "\\"Version\\": \"2012-10-17\", " +
    "\\"Statement\\": [{" +
        "\\"Effect\\": \"Allow\", " +
        "\\"Principal\\": {" +
            $" \\"AWS\\": \\"{userArn}\\" " +
        "}, " +
        "\\"Action\\": \\"sts:AssumeRole\"\\" " +
    "}]" +
"}";

// Permissions to list all buckets.
string policyDocument = "{" +
    "\\"Version\\": \"2012-10-17\", " +
    "\\"Statement\\": [{" +
        "\\"Action\\": \\"s3>ListAllMyBuckets\\\", " +
        "\\"Effect\\": \"Allow\", " +
        "\\"Resource\\": \\"*\"\\" " +
    "}]" +
"}";

// Create the role to allow listing the S3 buckets. Role names are
// not case sensitive and must be unique to the account for which it
// is created.
var role = await CreateRoleAsync(client, RoleName,
assumeRolePolicyDocument);
var roleArn = role.Arn;

// Create a policy with permissions to list S3 buckets
var policy = await CreatePolicyAsync(client, S3PolicyName, policyDocument);

// Wait 15 seconds for the policy to be created.
WaitABit(15, "Waiting for the policy to be available.");

// Attach the policy to the role you created earlier.
await AttachRoleAsync(client, policy.Arn, RoleName);

// Wait 15 seconds for the role to be updated.
Console.WriteLine();
WaitABit(15, "Waiting to time for the policy to be attached.");
```

```
// Use the AWS Security Token Service (AWS STS) to have the user
// assume the role we created.
var stsClient = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

// Wait for the new credentials to become valid.
WaitABit(10, "Waiting for the credentials to be valid.");

var assumedRoleCredentials = await AssumeS3RoleAsync(stsClient, "temporary-
session", roleArn);

// Try again to list the buckets using the client created with
// the new user's credentials. This time, it should work.
var s3Client2 = new AmazonS3Client(assumedRoleCredentials);

await ListMyBucketsAsync(s3Client2);

// Now clean up all the resources used in the example.
await DeleteResourcesAsync(client, accessKeyId, UserName, policy.Arn,
RoleName);

Console.WriteLine("IAM Demo completed.");
}

/// <summary>
/// Create a new IAM user.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="userName">A string representing the user name of the
new user.</param>
/// <returns>The newly created user.</returns>
public static async Task<User> CreateUserAsync(
    AmazonIdentityManagementServiceClient client,
    string userName)
{
    var request = new CreateUserRequest
    {
        UserName = userName,
    };

    var response = await client.CreateUserAsync(request);

    return response.User;
}

/// <summary>
/// Create a new AccessKey for the user.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="userName">The name of the user for whom to create the key.</
param>
/// <returns>A new IAM access key for the user.</returns>
public static async Task<AccessKey> CreateAccessKeyAsync(
    AmazonIdentityManagementServiceClient client,
    string userName)
{
    var request = new CreateAccessKeyRequest
    {
        UserName = userName,
    };

    var response = await client.CreateAccessKeyAsync(request);
```

```
        if (response.AccessKey is not null)
    {
        Console.WriteLine($"Successfully created Access Key for {userName}.");
    }

    return response.AccessKey;
}

/// <summary>
/// Create a policy to allow a user to list the buckets in an account.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="policyName">The name of the policy to create.</param>
/// <param name="policyDocument">The permissions policy document.</param>
/// <returns>The newly created ManagedPolicy object.</returns>
public static async Task<ManagedPolicy> CreatePolicyAsync(
    AmazonIdentityManagementServiceClient client,
    string policyName,
    string policyDocument)
{
    var request = new CreatePolicyRequest
    {
        PolicyName = policyName,
        PolicyDocument = policyDocument,
    };

    var response = await client.CreatePolicyAsync(request);

    return response.Policy;
}

/// <summary>
/// Attach the policy to the role so that the user can assume it.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="policyArn">The ARN of the policy to attach.</param>
/// <param name="roleName">The name of the role to attach the policy to.</param>
public static async Task AttachRoleAsync(
    AmazonIdentityManagementServiceClient client,
    string policyArn,
    string roleName)
{
    var request = new AttachRolePolicyRequest
    {
        PolicyArn = policyArn,
        RoleName = roleName,
    };

    var response = await client.AttachRolePolicyAsync(request);

    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine("Successfully attached the policy to the role.");
    }
    else
    {
        Console.WriteLine("Could not attach the policy.");
    }
}
```

```

    ///<summary>
    /// Create a new IAM role which we can attach to a user.
    ///</summary>
    ///<param name="client">The initialized IAM client object.</param>
    ///<param name="roleName">The name of the IAM role to create.</param>
    ///<param name="rolePermissions">The permissions which the role will have.</param>
    ///<returns>A Role object representing the newly created role.</returns>
    public static async Task<Role> CreateRoleAsync(
        AmazonIdentityManagementServiceClient client,
        string roleName,
        string rolePermissions)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePermissions,
        };

        var response = await client.CreateRoleAsync(request);

        return response.Role;
    }

    ///<summary>
    /// List the Amazon S3 buckets owned by the user.
    ///</summary>
    ///<param name="accessKeyId">The access key Id for the user.</param>
    ///<param name="secretAccessKey">The Secret access key for the user.</param>
    public static async Task ListMyBucketsAsync(AmazonS3Client client)
    {
        Console.WriteLine("\nPress <Enter> to list the S3 buckets using the new
user.\n");
        Console.ReadLine();

        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await client.ListBucketsAsync();

            // Loop through the list and print each bucket's name
            // and creation date.
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Listing S3 buckets:\n");
            response.Buckets
                .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
created on: {b.CreationDate}"));
        }
        catch (AmazonS3Exception ex)
        {
            // Something else went wrong. Display the error message.
            Console.WriteLine($"Error: {ex.Message}");
        }

        Console.WriteLine("Press <Enter> to continue.");
        Console.ReadLine();
    }

    ///<summary>
    /// Have the user assume the role that allows the role to be used to
    /// list all S3 buckets.

```

```

    ///</summary>
    ///<param name="client">An initialized AWS STS client object.</param>
    ///<param name="roleSession">The name of the session where the role
    /// assumption will be active.</param>
    ///<param name="roleToAssume">The Amazon Resource Name (ARN) of the
    /// role to assume.</param>
    ///<returns>The AssumedRoleUser object needed to perform the list
    /// buckets procedure.</returns>
    public static async Task<Credentials> AssumeS3RoleAsync(
        AmazonSecurityTokenServiceClient client,
        string roleSession,
        string roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };

        var response = await client.AssumeRoleAsync(request);

        return response.Credentials;
    }

    ///<summary>
    /// Delete the user, and other resources created for this example.
    ///</summary>
    ///<param name="client">The initialized client object.</param>
    ///<param name="accessKeyId">The Id of the user's access key.</param>
    ///<param name="userName">The user name of the user to delete.</param>
    ///<param name="policyName">The name of the policy to delete.</param>
    ///<param name="policyArn">The Amazon Resource Name ARN of the Policy to
    delete.</param>
    ///<param name="roleName">The name of the role that will be deleted.</param>
    public static async Task DeleteResourcesAsync(
        AmazonIdentityManagementServiceClient client,
        string accessKeyId,
        string userName,
        string policyArn,
        string roleName)
    {
        var detachPolicyResponse = await client.DetachRolePolicyAsync(new
DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        var delPolicyResponse = await client.DeletePolicyAsync(new
DeletePolicyRequest
        {
            PolicyArn = policyArn,
        });

        var delRoleResponse = await client.DeleteRoleAsync(new DeleteRoleRequest
        {
            RoleName = roleName,
        });

        var delAccessKey = await client.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
        });
    }
}

```

```
        UserName = userName,
    });

    var delUserResponse = await client.DeleteUserAsync(new DeleteUserRequest
    {
        UserName = userName,
    });

}

/// <summary>
/// Display a countdown and wait for a number of seconds.
/// </summary>
/// <param name="numSeconds">The number of seconds to wait.</param>
public static void WaitABit(int numSeconds, string msg)
{
    Console.WriteLine(msg);

    // Wait for the requested number of seconds.
    for (int i = numSeconds; i > 0; i--)
    {
        System.Threading.Thread.Sleep(1000);
        Console.Write($"{i}...");
    }

    Console.WriteLine("\n\nPress <Enter> to continue.");
    Console.ReadLine();
}

/// <summary>
/// Shows the a description of the features of the program.
/// </summary>
public static void DisplayInstructions()
{
    var separator = new string('-', 80);

    Console.WriteLine(separator);
    Console.WriteLine("IAM Basics");
    Console.WriteLine("This application uses the basic features of the AWS Identity and Access");
    Console.WriteLine("Management (IAM) creating, managing, and controlling access to resources for");
    Console.WriteLine("users. The application was created using the AWS SDK for .NET version 3.7 and");
    Console.WriteLine(".NET Core 5. The application performs the following actions:");
    Console.WriteLine();
    Console.WriteLine("1. Creates a user with no permissions");
    Console.WriteLine("2. Creates a role and policy that grants s3>ListAllMyBuckets permission");
    Console.WriteLine("3. Grants the user permission to assume the role");
    Console.WriteLine("4. Creates an Amazon Simple Storage Service (Amazon S3) client and tries");
    Console.WriteLine("    to list buckets. (This should fail.)");
    Console.WriteLine("5. Gets temporary credentials by assuming the role.");
    Console.WriteLine("6. Creates an Amazon S3 client object with the temporary credentials and");
    Console.WriteLine("    lists the buckets. (This time it should work.)");
    Console.WriteLine("7. Deletes all of the resources created.");
    Console.WriteLine(separator);
    Console.WriteLine("Press <Enter> to continue.");
    Console.ReadLine();
}
}
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## AWS KMS examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with AWS Key Management Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2409\)](#)

## Actions

### Create a grant for a key

The following code example shows how to create a grant for a KMS key.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

This examples grants a user permission to use a key for encryption and decryption.

```
public static async Task Main()
{
    var client = new AmazonKeyManagementServiceClient();

    // The identity that is given permission to perform the operations
    // specified in the grant.
    var grantee = "arn:aws:iam::111122223333:role/ExampleRole";

    // The identifier of the AWS KMS key to which the grant applies. You
    // can use the key ID or the Amazon Resource Name (ARN) of the KMS key.
    var keyId = "7c9ecc2-38cb-4c4f-9db3-766ee8dd3ad4";
```

```
var request = new CreateGrantRequest
{
    GranteePrincipal = grantee,
    KeyId = keyId,

    // A list of operations that the grant allows.
    Operations = new List<string>
    {
        "Encrypt",
        "Decrypt",
    },
};

var response = await client.CreateGrantAsync(request);

string grantId = response.GrantId; // The unique identifier of the grant.
string grantToken = response.GrantToken; // The grant token.

Console.WriteLine($"Id: {grantId}, Token: {grantToken}");
}
```

- For API details, see [CreateGrant](#) in *AWS SDK for .NET API Reference*.

## Create a key

The following code example shows how to create an AWS KMS key.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

public class CreateKey
{
    public static async Task Main()
    {
        // Note that if you need to create a Key in an AWS Region
        // other than the Region defined for the default user, you need to
        // pass the Region to the client constructor.
        var client = new AmazonKeyManagementServiceClient();

        // The call to CreateKeyAsync will create a symmetrical AWS KMS
        // key. For more information about symmetrical and asymmetrical
        // keys, see:
        //
        // https://docs.aws.amazon.com/kms/latest/developerguide/symm-asymm-
choose.html
        var response = await client.CreateKeyAsync(new CreateKeyRequest());

        // The KeyMetadata object contains information about the new AWS KMS key.
        KeyMetadata keyMetadata = response.KeyMetadata;
```

```
        if (keyMetadata is not null)
    {
        Console.WriteLine($"KMS Key: {keyMetadata.KeyId} was successfully
created.");
    }
    else
    {
        Console.WriteLine("Could not create KMS Key.");
    }
}

}
```

- For API details, see [CreateKey](#) in *AWS SDK for .NET API Reference*.

## Create an alias for a key

The following code example shows how to create an alias for a KMS key key.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

public class CreateAlias
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The alias name must start with alias/ and can be
        // up to 256 alphanumeric characters long.
        var aliasName = "alias/ExampleAlias";

        // The value supplied as the TargetKeyId can be either
        // the key ID or key Amazon Resource Name (ARN) of the
        // AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new CreateAliasRequest
        {
            AliasName = aliasName,
            TargetKeyId = keyId,
        };

        var response = await client.CreateAliasAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Alias, {aliasName}, successfully created.");
        }
        else
        {
            Console.WriteLine($"Could not create alias.");
        }
    }
}
```

```
        }  
    }  
}
```

- For API details, see [CreateAlias](#) in *AWS SDK for .NET API Reference*.

## Describe a key

The following code example shows how to describe a KMS key.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Threading.Tasks;  
using Amazon.KeyManagementService;  
using Amazon.KeyManagementService.Model;  
  
public class DescribeKey  
{  
    public static async Task Main()  
    {  
        var keyId = "7c9ecc2-38cb-4c4f-9db3-766ee8dd3ad4";  
        var request = new DescribeKeyRequest  
        {  
            KeyId = keyId,  
        };  
  
        var client = new AmazonKeyManagementServiceClient();  
  
        var response = await client.DescribeKeyAsync(request);  
        var metadata = response.KeyMetadata;  
  
        Console.WriteLine($"{{metadata.KeyId}} created on: {{metadata.CreationDate}}");  
        Console.WriteLine($"State: {{metadata.KeyState}}");  
        Console.WriteLine($"{{metadata.Description}}");  
    }  
}
```

- For API details, see [DescribeKey](#) in *AWS SDK for .NET API Reference*.

## Disable a key

The following code example shows how to disable a KMS key.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
```

```
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

public class DisableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to disable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";

        var request = new DisableKeyRequest
        {
            KeyId = keyId,
        };

        var response = await client.DisableKeyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            // Retrieve information about the key to show that it has now
            // been disabled.
            var describeResponse = await client.DescribeKeyAsync(new
DescribeKeyRequest
{
    KeyId = keyId,
});
            Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:
{describeResponse.KeyMetadata.KeyState}");
        }
    }
}
```

- For API details, see [DisableKey](#) in *AWS SDK for .NET API Reference*.

## Enable a key

The following code example shows how to enable a KMS key.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

public class EnableKey
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();

        // The identifier of the AWS KMS key to enable. You can use the
```

```
// key Id or the Amazon Resource Name (ARN) of the AWS KMS key.  
var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";  
  
var request = new EnableKeyRequest  
{  
    KeyId = keyId,  
};  
  
var response = await client.EnableKeyAsync(request);  
if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)  
{  
    // Retrieve information about the key to show that it has now  
    // been enabled.  
    var describeResponse = await client.DescribeKeyAsync(new  
DescribeKeyRequest  
{  
    KeyId = keyId,  
});  
    Console.WriteLine($"{describeResponse.KeyMetadata.KeyId} - state:  
{describeResponse.KeyMetadata.KeyState}");  
}  
}  
}
```

- For API details, see [EnableKey](#) in *AWS SDK for .NET API Reference*.

## List aliases for a key

The following code example shows how to list aliases for a KMS key.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Threading.Tasks;  
using Amazon.KeyManagementService;  
using Amazon.KeyManagementService.Model;  
  
public class ListAliases  
{  
    public static async Task Main()  
    {  
        var client = new AmazonKeyManagementServiceClient();  
        var request = new ListAliasesRequest();  
        var response = new ListAliasesResponse();  
  
        do  
        {  
            response = await client.ListAliasesAsync(request);  
  
            responseAliases.ForEach(alias =>  
            {  
                Console.WriteLine($"Created: {alias.CreationDate} Last Update:  
{alias.LastUpdatedDate} Name: {alias.AliasName}");  
            });  
  
            request.Marker = response.NextMarker;  
        } while (response.NextMarker != null);  
    }  
}
```

```
        }
        while (response.Truncated);
    }
}
```

- For API details, see [ListAliases](#) in *AWS SDK for .NET API Reference*.

## List grants for a key

The following code example shows how to list grants for a KMS key.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

public class ListGrants
{
    public static async Task Main()
    {
        // The identifier of the AWS KMS key to disable. You can use the
        // key Id or the Amazon Resource Name (ARN) of the AWS KMS key.
        var keyId = "1234abcd-12ab-34cd-56ef-1234567890ab";
        var client = new AmazonKeyManagementServiceClient();
        var request = new ListGrantsRequest
        {
            KeyId = keyId,
        };

        var response = new ListGrantsResponse();

        do
        {
            response = await client.ListGrantsAsync(request);

            response.Grants.ForEach(grant =>
            {
                Console.WriteLine($"{grant.GrantId}");
            });

            request.Marker = response.NextMarker;
        }
        while (response.Truncated);
    }
}
```

- For API details, see [ListGrants](#) in *AWS SDK for .NET API Reference*.

## List keys

The following code example shows how to list KMS keys.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.KeyManagementService;
using Amazon.KeyManagementService.Model;

public class ListKeys
{
    public static async Task Main()
    {
        var client = new AmazonKeyManagementServiceClient();
        var request = new ListKeysRequest();
        var response = new ListKeysResponse();

        do
        {
            response = await client.ListKeysAsync(request);

            response.Keys.ForEach(key =>
            {
                Console.WriteLine($"ID: {key.KeyId}, {key.KeyArn}");
            });

            // Set the Marker property when response.Truncated is true
            // in order to get the next keys.
            request.Marker = response.NextMarker;
        }
        while (response.Truncated);
    }
}
```

- For API details, see [ListKeys](#) in *AWS SDK for .NET API Reference*.

## Kinesis examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon Kinesis.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2416\)](#)

## Actions

### Add tags

The following code example shows how to add tags to a Kinesis stream.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// This example shows how to apply key/value pairs to an Amazon Kinesis
/// stream. The example was created using the AWS SDK for .NET version 3.7
/// and .NET Core 5.0.
/// </summary>
public class TagStream
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();

        string streamName = "AmazonKinesisStream";
        var tags = new Dictionary<string, string>
        {
            { "Project", "Sample Kinesis Project" },
            { "Application", "Sample Kinesis App" },
        };

        var success = await ApplyTagsToStreamAsync(client, streamName, tags);

        if (success)
        {
            Console.WriteLine($"Tags successfully added to {streamName}.");
        }
        else
        {
            Console.WriteLine("Tags were not added to the stream.");
        }
    }

    /// <summary>
    /// Applies the set of tags to the named Kinesis stream.
    /// </summary>
    /// <param name="client">The initialized Kinesis client.</param>
    /// <param name="streamName">The name of the Kinesis stream to which
    /// the tags will be attached.</param>
    /// <param name="tags">A sictionary containing key/value pairs which
    /// will be used to create the Kinesis tags.</param>
    /// <returns>A Boolean value which represents the success or failure
    /// of AddTagsToStreamAsync.</returns>
    public static async Task<bool> ApplyTagsToStreamAsync(
        IAmazonKinesis client,
        string streamName,
        Dictionary<string, string> tags)
    {
        var request = new AddTagsToStreamRequest
        {
            StreamName = streamName,
            Tags = tags,
        };

        var response = await client.AddTagsToStreamAsync(request);
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

- For API details, see [AddTagsToStream](#) in *AWS SDK for .NET API Reference*.

## Create a stream

The following code example shows how to create a Kinesis stream.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// This example shows how to create a new Amazon Kinesis stream. The
/// example was created using AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class CreateStream
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();

        string streamName = "AmazonKinesisStream";
        int shardCount = 1;

        var success = await CreateNewStreamAsync(client, streamName, shardCount);
        if (success)
        {
            Console.WriteLine($"The stream, {streamName} successfully created.");
        }
    }

    /// <summary>
    /// Creates a new Kinesis stream.
    /// </summary>
    /// <param name="client">An initialized Kinesis client.</param>
    /// <param name="streamName">The name for the new stream.</param>
    /// <param name="shardCount">The number of shards the new stream will
    /// use. The throughput of the stream is a function of the number of
    /// shards; more shards are required for greater provisioned
    /// throughput.</param>
    /// <returns>A Boolean value indicating whether the stream was created.</returns>
    public static async Task<bool> CreateNewStreamAsync(IAmazonKinesis client,
    string streamName, int shardCount)
    {
        var request = new CreateStreamRequest
        {
            StreamName = streamName,
            ShardCount = shardCount,
```

```
        };

        var response = await client.CreateStreamAsync(request);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

- For API details, see [CreateStream](#) in *AWS SDK for .NET API Reference*.

## Delete a stream

The following code example shows how to delete a Kinesis stream.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// Shows how to delete an Amazon Kinesis stream. The example was created
/// using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DeleteStream
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();
        string streamName = "AmazonKinesisStream";

        var success = await DeleteStreamAsync(client, streamName);

        if (success)
        {
            Console.WriteLine($"Stream, {streamName} successfully deleted.");
        }
        else
        {
            Console.WriteLine("Stream not deleted.");
        }
    }

    /// <summary>
    /// Deletes a Kinesis stream.
    /// </summary>
    /// <param name="client">An initialized Kinesis client object.</param>
    /// <param name="streamName">The name of the string to delete.</param>
    /// <returns>A Boolean value representing the success of the operation.</returns>
    public static async Task<bool> DeleteStreamAsync(IAmazonKinesis client, string
streamName)
    {
        // If EnforceConsumerDeletion is true, any consumers
        // of this stream will also be deleted. If it is set
```

```
// to false and this stream has any consumers, the
// call will fail with a ResourceInUseException.
var request = new DeleteStreamRequest
{
    StreamName = streamName,
    EnforceConsumerDeletion = true,
};

var response = await client.DeleteStreamAsync(request);

return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [DeleteStream](#) in *AWS SDK for .NET API Reference*.

## Deregister a consumer

The following code example shows how to deregister a consumer from a Kinesis stream.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// Shows how to deregister a consumer from an Amazon Kinesis stream. The
/// example was written using the AWS SDK for .NET 3.7 and .NET Core 5.0.
/// </summary>
public class DeregisterConsumer
{
    public static async Task Main(string[] args)
    {
        IAmazonKinesis client = new AmazonKinesisClient();

        string streamARN = "arn:aws:kinesis:us-west-2:000000000000:stream/
AmazonKinesisStream";
        string consumerName = "CONSUMER_NAME";
        string consumerARN = "arn:aws:kinesis:us-west-2:000000000000:stream/
AmazonKinesisStream/consumer/CONSUMER_NAME:000000000000";

        var success = await DeregisterConsumerAsync(client, streamARN, consumerARN,
consumerName);

        if (success)
        {
            Console.WriteLine($"{consumerName} successfully deregistered.");
        }
        else
        {
            Console.WriteLine($"{consumerName} was not successfully
deregistered.");
        }
    }
}
```

```
    /// <summary>
    /// Deregisters a consumer from a Kinesis stream.
    /// </summary>
    /// <param name="client">An initialized Kinesis client object.</param>
    /// <param name="streamARN">The ARN of a Kinesis stream.</param>
    /// <param name="consumerARN">The ARN of the consumer.</param>
    /// <param name="consumerName">The name of the consumer.</param>
    /// <returns>A Boolean value representing the success of the operation.</returns>
public static async Task<bool> DeregisterConsumerAsync(
    IAmazonKinesis client,
    string streamARN,
    string consumerARN,
    string consumerName)
{
    var request = new DeregisterStreamConsumerRequest
    {
        StreamARN = streamARN,
        ConsumerARN = consumerARN,
        ConsumerName = consumerName,
    };

    var response = await client.DeregisterStreamConsumerAsync(request);

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [DeregisterStreamConsumer](#) in *AWS SDK for .NET API Reference*.

## List streams

The following code example shows how to list information about one or more Kinesis streams.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

    /// <summary>
    /// Retrieves and displays a list of existing Amazon Kinesis streams. The
    /// example uses the AWS SDK for .NET version 3.7 and .NET Core 5.0.
    /// </summary>
public class ListStreams
{
    public static async Task Main(string[] args)
    {
        IAmazonKinesis client = new AmazonKinesisClient();
        var response = await client.ListStreamsAsync(new ListStreamsRequest());

        List<string> streamNames = response.StreamNames;
```

```
        if (streamNames.Count > 0)
    {
        streamNames
            .ForEach(s => Console.WriteLine($"Stream name: {s}"));
    }
    else
    {
        Console.WriteLine("No streams were found.");
    }
}
```

- For API details, see [ListStreams](#) in *AWS SDK for .NET API Reference*.

## List tags

The following code example shows how to list the tags associated with a Kinesis stream.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

///<summary>
/// Shows how to list the tags that have been attached to an Amazon Kinesis
/// stream. The example was created using the AWS SDK for .NET version 3.7
/// and .NET Core 5.0.
///</summary>
public class ListTags
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();
        string streamName = "AmazonKinesisStream";

        await ListTagsAsync(client, streamName);
    }

    ///<summary>
    /// List the tags attached to a Kinesis stream.
    ///</summary>
    ///<param name="client">An initialized Kinesis client object.</param>
    ///<param name="streamName">The name of the Kinesis stream for which you
    /// wish to display tags.</param>
    public static async Task ListTagsAsync(IAmazonKinesis client, string
streamName)
    {
        var request = new ListTagsForStreamRequest
        {
            StreamName = streamName,
            Limit = 10,
        };
    }
}
```

```
        var response = await client.ListTagsForStreamAsync(request);
        DisplayTags(response.Tags);

        while (response.HasMoreTags)
        {
            request.ExclusiveStartTagKey = response.Tags[response.Tags.Count - 1].Key;
            response = await client.ListTagsForStreamAsync(request);
        }
    }

    ///<summary>
    ///<summary> Displays the items in a list of Kinesis tags.
    ///</summary>
    ///<param name="tags">A list of the Tag objects to be displayed.</param>
    public static void DisplayTags(List<Tag> tags)
    {
        tags
            .ForEach(t => Console.WriteLine($"Key: {t.Key} Value: {t.Value}"));
    }
}
```

- For API details, see [ListTagsForStream](#) in *AWS SDK for .NET API Reference*.

## List the consumers of a stream

The following code example shows how to list the consumers of a Kinesis stream.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

    ///<summary>
    ///<summary> List the consumers of an Amazon Kinesis stream. This example was
    ///<summary> created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
    ///</summary>
    public class ListConsumers
    {
        public static async Task Main()
        {
            IAmazonKinesis client = new AmazonKinesisClient();

            string streamARN = "arn:aws:kinesis:us-east-2:000000000000:stream/
AmazonKinesisStream";
            int maxResults = 10;

            var consumers = await ListConsumersAsync(client, streamARN, maxResults);

            if (consumers.Count > 0)
            {
                consumers
                    .ForEach(c => Console.WriteLine($"Name: {c.ConsumerName} ARN:
{c.ConsumerARN}"));
            }
        }
    }
```

```
        }
    else
    {
        Console.WriteLine("No consumers found.");
    }
}

/// <summary>
/// Retrieve a list of the consumers for a Kinesis stream.
/// </summary>
/// <param name="client">An initialized Kinesis client object.</param>
/// <param name="streamARN">The ARN of the stream for which we want to
/// retrieve a list of clients.</param>
/// <param name="maxResults">The maximum number of results to return.</param>
/// <returns>A list of Consumer objects.</returns>
public static async Task<List<Consumer>> ListConsumersAsync(IAmazonKinesis
client, string streamARN, int maxResults)
{
    var request = new ListStreamConsumersRequest
    {
        StreamARN = streamARN,
        MaxResults = maxResults,
    };

    var response = await client.ListStreamConsumersAsync(request);

    return response.Consumers;
}
}
```

- For API details, see [ListStreamConsumers](#) in *AWS SDK for .NET API Reference*.

## Register a consumer

The following code example shows how to register a consumer to a Kinesis stream.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Kinesis;
using Amazon.Kinesis.Model;

/// <summary>
/// This example shows how to register a consumer to an Amazon Kinesis
/// stream. The example was written using the AWS SDK for .NET version 3.7
/// and .NET Core 5.0.
/// </summary>
public class RegisterConsumer
{
    public static async Task Main()
    {
        IAmazonKinesis client = new AmazonKinesisClient();
        string consumerName = "NEW_CONSUMER_NAME";
        string streamARN = "arn:aws:kinesis:us-east-2:000000000000:stream/
AmazonKinesisStream";
```

```
        var consumer = await RegisterConsumerAsync(client, consumerName,
streamARN);

        if (consumer is not null)
{
    Console.WriteLine($"{consumer.ConsumerName}");
}

/// <summary>
/// Registers the consumer to a Kinesis stream.
/// </summary>
/// <param name="client">The initialized Kinesis client object.</param>
/// <param name="consumerName">A string representing the consumer.</param>
/// <param name="streamARN">The ARN of the stream.</param>
/// <returns>A Consumer object that contains information about the consumer.</returns>
public static async Task<Consumer> RegisterConsumerAsync(IAmazonKinesis client,
string consumerName, string streamARN)
{
    var request = new RegisterStreamConsumerRequest
    {
        ConsumerName = consumerName,
        StreamARN = streamARN,
    };

    var response = await client.RegisterStreamConsumerAsync(request);
    return response.Consumer;
}
```

- For API details, see [RegisterStreamConsumer](#) in *AWS SDK for .NET API Reference*.

## Lambda examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with AWS Lambda.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2425\)](#)
- [Scenarios \(p. 2432\)](#)

## Actions

### Create a function

The following code example shows how to create a Lambda function.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Creates a new Lambda function.
/// </summary>
/// <param name="client">The initialized Lambda client object.</param>
/// <param name="functionName">The name of the function.</param>
/// <param name="s3Bucket">The S3 bucket where the zip file containing
/// the code is located.</param>
/// <param name="s3Key">The Amazon S3 key of the zip file.</param>
/// <param name="role">A role with the appropriate Lambda
/// permissions.</param>
/// <param name="handler">The name of the handler function.</param>
/// <returns>The Amazon Resource Name (ARN) of the newly created
/// Lambda function.</returns>
public async Task<string> CreateLambdaFunction(
    AmazonLambdaClient client,
    string functionName,
    string s3Bucket,
    string s3Key,
    string role,
    string handler)
{
    var functionCode = new FunctionCode
    {
        S3Bucket = s3Bucket,
        S3Key = s3Key,
    };

    var createFunctionRequest = new CreateFunctionRequest
    {
        FunctionName = functionName,
        Description = "Created by the Lambda .NET API",
        Code = functionCode,
        Handler = handler,
        Runtime = Runtime.Dotnet6,
        Role = role,
    };

    var reponse = await client.CreateFunctionAsync(createFunctionRequest);
    return reponse.FunctionArn;
}
```

- For API details, see [CreateFunction](#) in *AWS SDK for .NET API Reference*.

## Delete a function

The following code example shows how to delete a Lambda function.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Deletes an AWS Lambda function.
/// </summary>
/// <param name="client">An initialized Lambda client object.</param>
```

```
/// <param name="functionName">The name of the Lambda function to
/// delete.</param>
/// <returns>A Boolean value that indicates where the function was
/// successfully deleted.</returns>
public async Task<bool> DeleteLambdaFunction(AmazonLambdaClient client, string
functionName)
{
    var request = new DeleteFunctionRequest
    {
        FunctionName = functionName,
    };

    var response = await client.DeleteFunctionAsync(request);

    // A return value of NoContent means that the request was processed
    // (in this case, the function was deleted, and the return value
    // is intentionally blank.
    return response.StatusCode == System.Net.HttpStatusCode.NoContent;
}
```

- For API details, see [DeleteFunction](#) in *AWS SDK for .NET API Reference*.

## Get a function

The following code example shows how to get a Lambda function.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Gets information about a Lambda function.
/// </summary>
/// <param name="client">The initialized Lambda client object.</param>
/// <param name="functionName">The name of the Lambda function for
/// which to retrieve information.</param>
/// <returns>A System Threading Task.</returns>
public async Task<FunctionConfiguration> GetFunction(AmazonLambdaClient client,
string functionName)
{
    var functionRequest = new GetFunctionRequest
    {
        FunctionName = functionName,
    };

    var response = await client.GetFunctionAsync(functionRequest);
    return response.Configuration;
}
```

- For API details, see [GetFunction](#) in *AWS SDK for .NET API Reference*.

## Invoke a function

The following code example shows how to invoke a Lambda function.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System.Threading.Tasks;
using Amazon.Lambda;
using Amazon.Lambda.Model;

/// <summary>
/// Shows how to invoke an existing Amazon Lambda Function from a C#
/// application. The example was created using the AWS SDK for .NET and
/// .NET Core 5.0.
/// </summary>
public class InvokeFunction
{
    /// <summary>
    /// Initializes the Lambda client and then invokes the Lambda Function
    /// called "CreateDDBTable" with the parameter "\"DDBWorkTable\""
    /// to
    /// create the table called DDBWorkTable.
    /// </summary>
    public static async Task Main()
    {
        IAmazonLambda client = new AmazonLambdaClient();
        string functionName = "CreateDDBTable";
        string invokeArgs = "\"DDBWorkTable\"";

        var response = await client.InvokeAsync(
            new InvokeRequest
            {
                FunctionName = functionName,
                Payload = invokeArgs,
                InvocationType = "Event",
            });
    }
}

/// <summary>
/// Invokes a Lambda function.
/// </summary>
/// <param name="client">An initialized Lambda client object.</param>
/// <param name="functionName">The name of the Lambda function to
/// invoke.</param>
/// <returns>A System Threading Task.</returns>
public async Task<string> InvokeFunctionAsync(
    AmazonLambdaClient client,
    string functionName,
    string parameters)
{
    var payload = parameters;
    var request = new InvokeRequest
    {
        FunctionName = functionName,
        Payload = payload,
    };

    var response = await client.InvokeAsync(request);
    MemoryStream stream = response.Payload;
    string returnValue = System.Text.Encoding.UTF8.GetString(stream.ToArray());
    return returnValue;
}
```

- For API details, see [Invoke in AWS SDK for .NET API Reference](#).

## List functions

The following code example shows how to list Lambda functions.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.Lambda;
using Amazon.Lambda.Model;

/// <summary>
/// This example shows two ways to list the AWS Lambda functions you have
/// created for your account. It will only list the functions within one
/// AWS Region at a time, however, so you need to pass the AWS Region you
/// are interested in to the Lambda client object constructor. This example
/// was created with the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class ListFunctions
{
    public static async Task Main()
    {
        // If the AWS Region you are interested in listing is the same as
        // the AWS Region defined for the default user, you don't have to
        // supply the RegionEndpoint constant to the constructor.
        IAmazonLambda client = new AmazonLambdaClient(RegionEndpoint.USEast2);

        // First use the ListFunctionsAsync method.
        var functions1 = await ListFunctionsAsync(client);

        DisplayFunctionList(functions1);

        // Get the list again using a Lambda client paginator.
        var functions2 = await ListFunctionsPaginatorAsync(client);

        DisplayFunctionList(functions2);
    }

    /// <summary>
    /// Calls the asynchronous ListFunctionsAsync method of the Lambda
    /// client to retrieve the list of functions in the AWS Region with
    /// which the Lambda client was initialized.
    /// </summary>
    /// <param name="client">The initialized Lambda client which will be
    /// used to retrieve the list of Lambda functions.</param>
    /// <returns>A list of Lambda functions configuration information.</returns>
    public static async Task<List<FunctionConfiguration>>
ListFunctionsAsync(IAmazonLambda client)
    {
        // Get the list of functions. The response will have a property
        // called Functions, a list of information about the Lambda
        // functions defined on your account in the specified Region.
        var response = await client.ListFunctionsAsync();
```

```
        return response.Functions;
    }

    /// <summary>
    /// Uses a Lambda paginator to retrieve the list of functions in the
    /// AWS Region with which the Lambda client was initialized.
    /// </summary>
    /// <param name="client">The initialized Lambda client which will be
    /// used to retrieve the list of Lambda functions.</param>
    /// <returns>A list of Lambda functions configuration information.</returns>
    public static async Task<List<FunctionConfiguration>>
ListFunctionsPaginatorAsync(IAmazonLambda client)
{
    Console.WriteLine("\nNow let's show the list using a paginator.\n");

    // Get the list of functions using a paginator.
    var paginator = client.Paginator.ListFunctions(new
ListFunctionsRequest());

    // Defined return a list of function information to the caller
    // for display using the DisplayFunctionList method.
    var functions = new List<FunctionConfiguration>();

    await foreach (var resp in paginator.Responses)
    {
        resp.Functions
            .ForEach(f => functions.Add(f));
    }

    return functions;
}

    /// <summary>
    /// Displays the details of each function in the list of functions
    /// passed to the method.
    /// </summary>
    /// <param name="functions">A list of FunctionConfiguration objects.</param>
    public static void DisplayFunctionList(List<FunctionConfiguration> functions)
{
    // Display a list of the Lambda functions on the console.
    functions
        .ForEach(f => Console.WriteLine($"{f.FunctionName}\t{f.Handler}"));
}
}

    /// <summary>
    /// Gets a list of Lambda functions.
    /// </summary>
    /// <param name="client">The initialized Lambda client object.</param>
    /// <returns>A list of FunctionConfiguration objects.</returns>
    public async Task<List<FunctionConfiguration>> ListFunctions(AmazonLambdaClient
client)
{
    var reponse = await client.ListFunctionsAsync();
    var functionList = reponse.Functions;
    return functionList;
}
```

List functions using a paginator.

```
/// <summary>
/// Uses a Lambda paginator to retrieve the list of functions in the
/// AWS Region with which the Lambda client was initialized.
/// </summary>
/// <param name="client">The initialized Lambda client which will be
/// used to retrieve the list of Lambda functions.</param>
/// <returns>A list of Lambda functions configuration information.</returns>
public static async Task<List<FunctionConfiguration>>
ListFunctionsPaginatorAsync(IAmazonLambda client)
{
    Console.WriteLine("\nNow let's show the list using a paginator.\n");

    // Get the list of functions using a paginator.
    var paginator = client.Paginator.ListFunctions(new
ListFunctionsRequest());

    // Defined return a list of function information to the caller
    // for display using the DisplayFunctionList method.
    var functions = new List<FunctionConfiguration>();

    await foreach (var resp in paginator.Responses)
    {
        resp.Functions
            .ForEach(f => functions.Add(f));
    }

    return functions;
}
```

- For API details, see [ListFunctions](#) in *AWS SDK for .NET API Reference*.

## Update function code

The following code example shows how to update Lambda function code.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Updates an existing Lambda function.
/// </summary>
/// <param name="client">An initialized Lambda client object.</param>
/// <param name="functionName">The name of the Lambda function to update.</param>
/// <param name="bucketName">The bucket where the zip file containing
/// the Lambda function code is stored.</param>
/// <param name="key">The key name of the source code file.</param>
/// <returns>A System.Threading.Task.</returns>
public async Task UpdateFunctionCode(
    AmazonLambdaClient client,
    string functionName,
    string bucketName,
    string key)
{
    var functionCodeRequest = new UpdateFunctionCodeRequest
```

```
        {
            FunctionName = functionName,
            Publish = true,
            S3Bucket = bucketName,
            S3Key = key,
        };

        var response = await client.UpdateFunctionCodeAsync(functionCodeRequest);
        Console.WriteLine($"The Function was last modified at
{response.LastModified}.");
    }
}
```

- For API details, see [UpdateFunctionCode](#) in *AWS SDK for .NET API Reference*.

## Update function configuration

The following code example shows how to update Lambda function configuration.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public async Task<bool> UpdateFunctionConfigurationAsync(
    AmazonLambdaClient client,
    string functionName,
    string functionHandler,
    Dictionary<string, string> environmentVariables)
{
    var request = new UpdateFunctionConfigurationRequest
    {
        Handler = functionHandler,
        FunctionName = functionName,
        Environment = new Amazon.Lambda.Model.Environment { Variables =
environmentVariables },
    };

    var response = await client.UpdateFunctionConfigurationAsync(request);

    Console.WriteLine(response.LastModified);

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for .NET API Reference*.

## Scenarios

### Get started with functions

The following code example shows how to:

- Create an AWS Identity and Access Management (IAM) role that grants Lambda permission to write to logs.
- Create a Lambda function and upload handler code.

- Invoke the function with a single parameter and get results.
- Update the function code and configure its Lambda environment with an environment variable.
- Invoke the function with new parameters and get results. Display the execution log that's returned from the invocation.
- List the functions for your account.
- Delete the IAM role and the Lambda function.

For more information, see [Create a Lambda function with the console](#).

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
global using Amazon;
global using Amazon.Lambda;
global using Amazon.Lambda.Model;
global using Amazon.IdentityManagement;
global using Amazon.IdentityManagement.Model;
global using Lambda_Basics;
global using Microsoft.Extensions.Configuration;

// The following variables will be loaded from a configuration file:
//
//   functionName - The name of the Lambda function.
//   roleName - The IAM service role that has Lambda permissions.
//   handler - The fully qualified method name (for example,
//             example.Handler::handleRequest).
//   bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name
//               that contains the .zip or .jar used to update the Lambda function's code.
//   key - The Amazon S3 key name that represents the .zip or .jar (for
//         example, LambdaHello-1.0-SNAPSHOT.jar).
//   keyUpdate - The Amazon S3 key name that represents the updated .zip (for
//             example, "updated-function.zip").

var configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

string functionName = configuration["FunctionName"];
string roleName = configuration["RoleName"];
string policyDocument = "{" +
    "\"Version\": \"2012-10-17\", " +
    "\"Statement\": [ " +
        "{" +
            "\"Effect\": \"Allow\", " +
            "\"Principal\": {\"+" +
                "\"Service\": \"lambda.amazonaws.com\" " +
            "}, " +
            "\"Action\": \"sts:AssumeRole\" " +
        "}" +
    "]," +
"}";

var incrementHandler = configuration["IncrementHandler"];
var calculatorHandler = configuration["CalculatorHandler"];
```

```
var bucketName = configuration["BucketName"];
var key = configuration["Key"];
var updateKey = configuration["UpdateKey"];

string sepBar = new('-', 80);

var lambdaClient = new AmazonLambdaClient();
var lambdaMethods = new LambdaMethods();
var lambdaRoleMethods = new LambdaRoleMethods();

ShowOverview();

// Create the policy to use with the Lambda functions and then attach the
// policy to a new role.
var roleArn = await lambdaRoleMethods.CreateLambdaRole(roleName, policyDocument);

Console.WriteLine("Waiting for role to become active.");
System.Threading.Thread.Sleep(10000);

// Create the Lambda function using a zip file stored in an S3 bucket.
Console.WriteLine(sepBar);
Console.WriteLine($"Creating the AWS Lambda function: {functionName}.");
var lambdaArn = await lambdaMethods.CreateLambdaFunction(
    lambdaClient,
    functionName,
    bucketName,
    key,
    roleArn,
    incrementHandler);

Console.WriteLine(sepBar);
Console.WriteLine($"The AWS Lambda ARN is {lambdaArn}");

// Get the Lambda function.
Console.WriteLine($"Getting the {functionName} AWS Lambda function.");
FunctionConfiguration config;
do
{
    config = await lambdaMethods.GetFunction(lambdaClient, functionName);
    Console.Write(".");
}
while (config.State != State.Active);

Console.WriteLine($"\\nThe function, {functionName} has been created.");
Console.WriteLine($"The runtime of this Lambda function is {config.Runtime}.");
PressEnter();

// List the Lambda functions.
Console.WriteLine(sepBar);
Console.WriteLine("Listing all Lambda functions.");
var functions = await lambdaMethods.ListFunctions(lambdaClient);
DisplayFunctionList(functions);
Console.WriteLine(sepBar);

Console.WriteLine(sepBar);
Console.WriteLine("Invoke the Lambda increment function.");
string? value;
do
{
    Console.Write("Enter a value to increment: ");
    value = Console.ReadLine();
}
while (value == string.Empty);

string functionParameters = "{" +
```

```
        "\"action\": \"increment\", " +
        "\\"x\": \"" + value + "\"" +
    "}";
var answer = await lambdaMethods.InvokeFunctionAsync(lambdaClient, functionName,
    functionParameters);
Console.WriteLine($"{value} + 1 = {answer}.");
Console.WriteLine(sepBar);
Console.WriteLine("Now update the Lambda function code.");
await lambdaMethods.UpdateFunctionCode(lambdaClient, functionName, bucketName,
    updateKey);

do
{
    config = await lambdaMethods.GetFunction(lambdaClient, functionName);
    Console.Write(".");
}
while (config.LastUpdateStatus == LastUpdateStatus.InProgress);

await lambdaMethods.UpdateFunctionConfigurationAsync(
    lambdaClient,
    functionName,
    configuration["CalculatorHandler"],
    new Dictionary<string, string> { { "LOG_LEVEL", "DEBUG" } });
do
{
    config = await lambdaMethods.GetFunction(lambdaClient, functionName);
    Console.Write(".");
}
while (config.LastUpdateStatus == LastUpdateStatus.InProgress);

Console.WriteLine();
Console.WriteLine(sepBar);
Console.WriteLine("Now call the updated function...");

// Get two numbers and an action from the user.
value = string.Empty;
do
{
    Console.Write("Enter the first value: ");
    value = Console.ReadLine();
}
while (value == string.Empty);

string? value2;
do
{
    Console.Write("Enter a second value: ");
    value2 = Console.ReadLine();
}
while (value2 == string.Empty);

string? opSelected;

Console.WriteLine("Select the operation to perform:");
Console.WriteLine("\t1. add");
Console.WriteLine("\t2. subtract");
Console.WriteLine("\t3. multiply");
Console.WriteLine("\t4. divide");
Console.WriteLine("Enter the number (1, 2, 3, or 4) of the operation you want to
    perform: ");
do
{
    Console.Write("Your choice? ");
    opSelected = Console.ReadLine();
```

```
        }

        while (opSelected == string.Empty);

        var operation = (opSelected) switch
        {
            "1" => "add",
            "2" => "subtract",
            "3" => "multiply",
            "4" => "divide",
            _ => "add",
        };

        functionParameters = "{" +
            "\"action\": \"\" + operation + "\", \"\" +
            "\"x\": \"\" + value + "\", \"\" +
            "\"y\": \"\" + value2 + \"\" +
        "}";

        answer = await lambdaMethods.InvokeFunctionAsync(lambdaClient, functionName,
            functionParameters);
        Console.WriteLine($"The answer when we {operation} the two numbers is: {answer}.");

        PressEnter();

        // Delete the function created earlier.
        Console.WriteLine(sepBar);
        Console.WriteLine("Delete the AWS Lambda function.");
        var success = await lambdaMethods.DeleteLambdaFunction(lambdaClient, functionName);
        if (success)
        {
            Console.WriteLine($"The {functionName} function was deleted.");
        }
        else
        {
            Console.WriteLine($"Could not remove the function {functionName}");
        }

        // Now delete the IAM role created for use with the functions
        // created by the application.
        success = await lambdaRoleMethods.DeleteLambdaRole(roleName);
        if (success)
        {
            Console.WriteLine("The role has been successfully removed.");
        }
        else
        {
            Console.WriteLine("Couldn't delete the role.");
        }

        Console.WriteLine("The Lambda Scenario is now complete.");
        PressEnter();

        // Displays a formatted list of existing functions returned by the
        // LambdaMethods.ListFunctions.
        void DisplayFunctionList(List<FunctionConfiguration> functions)
        {
            functions.ForEach(functionConfig =>
            {
                Console.WriteLine($"{functionConfig.FunctionName}\t{functionConfig.Description}");
            });
        }

        // Displays an overview of the application.
        void ShowOverview()
        {
```

```
Console.WriteLine("Welcome to the AWS Lambda Basics Example");
Console.WriteLine("Getting started with functions");
Console.WriteLine(sepBar);
Console.WriteLine("This scenario performs the following operations:");
Console.WriteLine("\t 1. Creates an IAM policy that will be used by AWS Lambda.");
Console.WriteLine("\t 2. Attaches the policy to a new IAM role.");
Console.WriteLine("\t 3. Creates an AWS Lambda function.");
Console.WriteLine("\t 4. Gets a specific AWS Lambda function.");
Console.WriteLine("\t 5. Lists all Lambda functions.");
Console.WriteLine("\t 6. Invokes the Lambda function.");
Console.WriteLine("\t 7. Updates the Lambda function's code.");
Console.WriteLine("\t 8. Updates the Lambda function's configuration.");
Console.WriteLine("\t 9. Invokes the updated function.");
Console.WriteLine("\t10. Deletes the Lambda function.");
Console.WriteLine("\t11. Deletes the IAM role.");
PressEnter();
}

// Wait for the user to press the Enter key.
void PressEnter()
{
    Console.Write("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine();
}
```

Define a Lambda handler that increments a number.

```
using Amazon.Lambda.Core;

// Assembly attribute to enable the Lambda function's JSON input to be converted into
// a .NET class.
[assembly:
LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer))]

namespace LambdaIncrement;

public class Function
{

    /// <summary>
    /// A simple function increments the integer parameter.
    /// </summary>
    /// <param name="input">A JSON string containing an action, which must be
    /// "increment" and a string representing the value to increment.</param>
    /// <param name="context">The context object passed by Lambda containing
    /// information about invocation, function, and execution environment.</param>
    /// <returns>A string representing the incremented value of the parameter.</returns>
    public int FunctionHandler(Dictionary<string, string> input, ILambdaContext
context)
    {
        if (input["action"] == "increment")
        {
            int inputValue = Convert.ToInt32(input["x"]);
            return inputValue + 1;
        }
        else
        {
            return 0;
        }
    }
}
```

Define a second Lambda handler that performs arithmetic operations.

```
using Amazon.Lambda.Core;
using System.Diagnostics;

// Assembly attribute to enable the Lambda function's JSON input to be converted into
// a .NET class.
[assembly:
LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer))]

namespace LambdaCalculator;

public class Function
{

    /// <summary>
    /// A simple function that takes two number in string format and performs
    /// the requested arithmetic function.
    /// </summary>
    /// <param name="input">JSON data containing an action, and x and y values.
    /// Valid actions include: add, subtract, multiply, and divide.</param>
    /// <param name="context">The context object passed by Lambda containing
    /// information about invocation, function, and execution environment.</param>
    /// <returns>A string representing the results of the calculation.</returns>
    public int FunctionHandler(Dictionary<string, string> input, ILambdaContext
context)
{
    var action = input["action"];
    int x = Convert.ToInt32(input["x"]);
    int y = Convert.ToInt32(input["y"]);
    int result;
    switch (action)
    {
        case "add":
            result = x + y;
            break;
        case "subtract":
            result = x - y;
            break;
        case "multiply":
            result = x * y;
            break;
        case "divide":
            if (y == 0)
            {
                Console.Error.WriteLine("Divide by zero error.");
                result = 0;
            }
            else
                result = x / y;
            break;
        default:
            Console.Error.WriteLine($"{action} is not a valid operation.");
            result = 0;
            break;
    }
    return result;
}
}
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## Organizations examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with AWS Organizations.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2439\)](#)

## Actions

### Attach a policy to a target

The following code example shows how to attach an Organizations policy to a target.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

///<summary>
/// Shows how to attach an AWS Organizations policy to an organization,
/// an organizational unit, or an account.
///</summary>
public class AttachPolicy
{
    ///<summary>
    /// Initializes the Organizations client object and then calls the
    /// AttachPolicyAsync method to attach the policy to the root
    /// organization.
    ///</summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
        var policyId = "p-00000000";
        var targetId = "r-0000";

        var request = new AttachPolicyRequest
```

```
{  
    PolicyId = policyId,  
    TargetId = targetId,  
};  
  
var response = await client.AttachPolicyAsync(request);  
  
if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)  
{  
    Console.WriteLine($"Successfully attached Policy ID {policyId} to  
Target ID: {targetId}.");  
}  
else  
{  
    Console.WriteLine("Was not successful in attaching the policy.");  
}  
}  
}
```

- For API details, see [AttachPolicy](#) in *AWS SDK for .NET API Reference*.

## Create a policy

The following code example shows how to create an Organizations policy.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Threading.Tasks;  
using Amazon.Organizations;  
using Amazon.Organizations.Model;  
  
///<summary>  
/// Creates a new AWS Organizations Policy. The example was created  
/// using the AWS SDK for .NET version 3.7 and .NET Core 5.0.  
///</summary>  
class CreatePolicy  
{  
    ///<summary>  
    /// Initializes the AWS Organizations client object, uses it to  
    /// create a new Organizations Policy, and then displays information  
    /// about the newly created Policy.  
    ///</summary>  
    static async Task Main()  
    {  
        IAmazonOrganizations client = new AmazonOrganizationsClient();  
        var policyContent = "{" +  
            " \"Version\": \"2012-10-17\", " +  
            " \"Statement\" : [{" +  
                " \"Action\" : ["s3:*"], " +  
                " \"Effect\" : \"Allow\", " +  
                " \"Resource\" : \"*\" " +  
            "}], " +  
        "};  
        try  
        {  
            var response = await client.CreatePolicyAsync(new CreatePolicyRequest
```

```
        {
            Content = policyContent,
            Description = "Enables admins of attached accounts to delegate all
Amazon S3 permissions",
            Name = "AllowAllS3Actions",
            Type = "SERVICE_CONTROL_POLICY",
        });

        Policy policy = response.Policy;
        Console.WriteLine($"{policy.PolicySummary.Name} has the following
content: {policy.Content}");
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
}
```

- For API details, see [CreatePolicy](#) in *AWS SDK for .NET API Reference*.

## Create an account

The following code example shows how to create an Organizations account.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

///<summary>
/// Creates a new AWS Organizations account. The example was created
/// using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
///</summary>
class CreateAccount
{
    ///<summary>
    /// Initializes an Organizations client object and uses it to create
    /// the new account with the name specified in accountName.
    ///</summary>
    static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();
        var accountName = "ExampleAccount";
        var email = "someone@example.com";

        var request = new CreateAccountRequest
        {
            AccountName = accountName,
            Email = email,
        };

        var response = await client.CreateAccountAsync(request);
        var status = response.CreateAccountStatus;

        Console.WriteLine($"The status of {status.AccountName} is {status.State}.");
    }
}
```

```
    }
```

- For API details, see [CreateAccount](#) in *AWS SDK for .NET API Reference*.

## Create an organization

The following code example shows how to create an Organizations organization.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

///<summary>
/// Creates an organization in AWS Organizations. The example was created
/// using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
///</summary>
public class CreateOrganization
{
    ///<summary>
    /// Creates an Organizations client object and then uses it to create
    /// a new organization with the default user as the administrator, and
    /// then displays information about the new organization.
    ///</summary>
    public static async Task Main()
    {
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var response = await client.CreateOrganizationAsync(new
CreateOrganizationRequest
        {
            FeatureSet = "ALL",
        });

        Organization newOrg = response.Organization;

        Console.WriteLine($"Organization: {newOrg.Id} Main Account:
{newOrg.MasterAccountId}");
    }
}
```

- For API details, see [CreateOrganization](#) in *AWS SDK for .NET API Reference*.

## Create an organizational unit

The following code example shows how to create an Organizations organizational unit.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

<:/// <summary>
<:/// Creates a new organizational unit in AWS Organizations. The example was
<:/// created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
<:/// </summary>
public class CreateOrganizationalUnit
{
    <:/// <summary>
    <:/// Initializes an Organizations client object and then uses it to call
    <:/// the CreateOrganizationalUnit method. If the call succeeds, it
    <:/// displays information about the new organizational unit.
    <:/// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var orgUnitName = "ProductDevelopmentUnit";

        var request = new CreateOrganizationalUnitRequest
        {
            Name = orgUnitName,
            ParentId = "r-0000",
        };

        var response = await client.CreateOrganizationalUnitAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully created organizational unit:
{orgUnitName}.");
            Console.WriteLine($"Organizational unit {orgUnitName} Details");
            Console.WriteLine($"ARN: {response.OrganizationalUnit.Arn} Id:
{response.OrganizationalUnit.Id}");
        }
        else
        {
            Console.WriteLine("Could not create new organizational unit.");
        }
    }
}
```

- For API details, see [CreateOrganizationalUnit in AWS SDK for .NET API Reference](#).

## Delete a policy

The following code example shows how to delete an Organizations policy.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
```

```
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Deletes an existing AWS Organizations policy. This example was
/// created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DeletePolicy
{
    /// <summary>
    /// Initializes the Organizations client object and then uses it to
    /// delete the policy with the specified policyId.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var policyId = "p-00000000";

        var request = new DeletePolicyRequest
        {
            PolicyId = policyId,
        };

        var response = await client.DeletePolicyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully deleted Policy: {policyId}.");
        }
        else
        {
            Console.WriteLine($"Could not delete Policy: {policyId}.");
        }
    }
}
```

- For API details, see [DeletePolicy](#) in *AWS SDK for .NET API Reference*.

## Delete an organization

The following code example shows how to delete an Organizations organization.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to delete an existing organization using the AWS
/// Organizations Service. This example was created using the AWS SDK for
/// .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DeleteOrganization
```

```
{  
    ///<summary>  
    /// Initializes the Organizations client and then calls  
    /// DeleteOrganizationAsync to delete the organization.  
    ///</summary>  
    public static async Task Main()  
    {  
        // Create the client object using the default account.  
        IAmazonOrganizations client = new AmazonOrganizationsClient();  
  
        var response = await client.DeleteOrganizationAsync(new  
DeleteOrganizationRequest());  
  
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)  
        {  
            Console.WriteLine("Successfully deleted organization.");  
        }  
        else  
        {  
            Console.WriteLine("Could not delete organization.");  
        }  
    }  
}
```

- For API details, see [DeleteOrganization](#) in *AWS SDK for .NET API Reference*.

## Delete an organizational unit

The following code example shows how to delete an Organizations organizational unit.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Threading.Tasks;  
using Amazon.Organizations;  
using Amazon.Organizations.Model;  
  
///<summary>  
/// Shows how to delete an existing AWS Organizations organizational unit.  
/// This example was created using the AWS SDK for .NET version 3.7 and  
/// .NET Core 5.0.  
///</summary>  
public class DeleteOrganizationalUnit  
{  
    ///<summary>  
    /// Initializes the Organizations client object and calls  
    /// DeleteOrganizationalUnitAsync to delete the organizational unit  
    /// with the selected ID.  
    ///</summary>  
    public static async Task Main()  
    {  
        // Create the client object using the default account.  
        IAmazonOrganizations client = new AmazonOrganizationsClient();  
  
        var orgUnitId = "ou-0000-00000000";  
  
        var request = new DeleteOrganizationalUnitRequest
```

```
{  
    OrganizationUnitId = orgUnitId,  
};  
  
var response = await client.DeleteOrganizationalUnitAsync(request);  
  
if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)  
{  
    Console.WriteLine($"Successfully deleted the organizational unit with  
ID: {orgUnitId}.");  
}  
else  
{  
    Console.WriteLine($"Could not delete the organizational unit with ID:  
{orgUnitId}.");  
}  
}  
}
```

- For API details, see [DeleteOrganizationalUnit in AWS SDK for .NET API Reference](#).

## Detach a policy from a target

The following code example shows how to detach an Organizations policy from a target.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Threading.Tasks;  
using Amazon.Organizations;  
using Amazon.Organizations.Model;  
  
/// <summary>  
/// Shows how to detach a policy from an AWS Organizations organization,  
/// organizational unit, or account. The example was creating using the  
/// AWS SDK for .NET 3.7 and .NET Core 5.0.  
/// </summary>  
public class DetachPolicy  
{  
    /// <summary>  
    /// Initializes the Organizations client object and uses it to call  
    /// DetachPolicyAsync to detach the policy.  
    /// </summary>  
    public static async Task Main()  
    {  
        // Create the client object using the default account.  
        IAmazonOrganizations client = new AmazonOrganizationsClient();  
  
        var policyId = "p-00000000";  
        var targetId = "r-0000";  
  
        var request = new DetachPolicyRequest  
        {  
            PolicyId = policyId,  
            TargetId = targetId,  
        };  
    }  
}
```

```
        var response = await client.DetachPolicyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"Successfully detached policy with Policy Id:
{policyId}.");
        }
        else
        {
            Console.WriteLine("Could not detach the policy.");
        }
    }
}
```

- For API details, see [DetachPolicy](#) in *AWS SDK for .NET API Reference*.

## List accounts

The following code example shows how to list accounts for an Organizations organization.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

///<summary>
/// Uses the AWS Organizations service to list the accounts associated
/// with the default account. The example was created using the AWS SDK for
/// .NET and .NET Core 5.0.
///</summary>
class ListAccounts
{
    ///<summary>
    /// Creates the Organizations client and then calls its
    /// ListAccountsAsync method.
    ///</summary>
    static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var request = new ListAccountsRequest
        {
            MaxResults = 5,
        };

        var response = new ListAccountsResponse();
        try
        {
            do
            {
                response = await client.ListAccountsAsync(request);
                response.Accounts.ForEach(a => DisplayAccounts(a));
                if (response.NextToken is not null)
                {

```

```
        request.NextToken = response.NextToken;
    }
} while (response.NextToken is not null);
}
catch (AWSOrganizationsNotInUseException ex)
{
    Console.WriteLine(ex.Message);
}

/// <summary>
/// Displays information about an Organizations account.
/// </summary>
/// <param name="account">An Organizations account for which to display
/// information on the console.</param>
private static void DisplayAccounts(Account account)
{
    string accountInfo = $"{account.Id} {account.Name}\t{account.Status}";

    Console.WriteLine(accountInfo);
}
}
```

- For API details, see [ListAccounts](#) in *AWS SDK for .NET API Reference*.

## List organizational units

The following code example shows how to list Organizations organizational units.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Lists the AWS Organizations organizational units that belong to an
/// organization. The example was created using the AWS SDK for .NET
/// version 3.7 and .NET Core 5.0.
/// </summary>
public class ListOrganizationalUnitsForParent
{
    /// <summary>
    /// Initializes the Organizations client object and then uses it to
    /// call the ListOrganizationalUnitsForParentAsync method to retrieve
    /// the list of organizational units.
    /// </summary>
    public static async Task Main()
    {
        // Create the client object using the default account.
        IAmazonOrganizations client = new AmazonOrganizationsClient();

        var parentId = "r-0000";

        var request = new ListOrganizationalUnitsForParentRequest
        {
            ParentId = parentId,
```

```
        MaxResults = 5,
    };

    var response = new ListOrganizationalUnitsForParentResponse();
    try
    {
        do
        {
            response = await
client.ListOrganizationalUnitsForParentAsync(request);
            response.OrganizationalUnits.ForEach(u =>
DisplayOrganizationalUnit(u));
            if (response.NextToken is not null)
            {
                request.NextToken = response.NextToken;
            }
        } while (response.NextToken is not null);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}

/// <summary>
/// Displays information about an Organizations organizational unit.
/// </summary>
/// <param name="unit">The OrganizationalUnit for which to display
/// information.</param>
public static void DisplayOrganizationalUnit(OrganizationalUnit unit)
{
    string accountInfo = $"{unit.Id} {unit.Name}\t{unit.ArN}";

    Console.WriteLine(accountInfo);
}
```

- For API details, see [ListOrganizationalUnitsForParent](#) in *AWS SDK for .NET API Reference*.

## List policies

The following code example shows how to list Organizations policies.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Organizations;
using Amazon.Organizations.Model;

/// <summary>
/// Shows how to list the AWS Organizations policies associated with an
/// organization. The example was created with the AWS SDK for .NET version
/// 3.7 and .NET Core 5.0.
/// </summary>
public class ListPolicies
{
```

```
/// <summary>
/// Initializes an Organizations client object, and then calls its
/// ListPoliciesAsync method.
/// </summary>
public static async Task Main()
{
    // Create the client object using the default account.
    IAmazonOrganizations client = new AmazonOrganizationsClient();

    // The value for the Filter parameter is required and must be
    // one of the following:
    // AISERVICES_OPT_OUT_POLICY
    // BACKUP_POLICY
    // SERVICE_CONTROL_POLICY
    // TAG_POLICY
    var request = new ListPoliciesRequest
    {
        Filter = "SERVICE_CONTROL_POLICY",
        MaxResults = 5,
    };

    var response = new ListPoliciesResponse();
    try
    {
        do
        {
            response = await client.ListPoliciesAsync(request);
            response.Policies.ForEach(p => DisplayPolicies(p));
            if (response.NextToken is not null)
            {
                request.NextToken = response.NextToken;
            }
        } while (response.NextToken is not null);
    }
    catch (AWSOrganizationsNotInUseException ex)
    {
        Console.WriteLine(ex.Message);
    }
}

/// <summary>
/// Displays information about the Organizations policies associated
/// with an organization.
/// </summary>
/// <param name="policy">An Organizations policy summary to display
/// information on the console.</param>
private static void DisplayPolicies(PolicySummary policy)
{
    string policyInfo = $"{policy.Id} {policy.Name}\t{policy.Description}";

    Console.WriteLine(policyInfo);
}
```

- For API details, see [ListPolicies](#) in *AWS SDK for .NET API Reference*.

## Amazon Polly examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon Polly.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2451\)](#)

## Actions

### Delete a lexicon

The following code example shows how to delete an Amazon Polly lexicon.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class DeleteLexicon
{
    public static async Task Main()
    {
        string lexiconName = "SampleLexicon";

        var client = new AmazonPollyClient();

        var success = await DeletePollyLexiconAsync(client, lexiconName);

        if (success)
        {
            Console.WriteLine($"Successfully deleted {lexiconName}.");
        }
        else
        {
            Console.WriteLine($"Could not delete {lexiconName}.");
        }
    }

    ///<summary>
    /// Deletes the named Amazon Polly lexicon.
    ///</summary>
    ///<param name="client">The initialized Amazon Polly client object.</param>
    ///<param name="lexiconName">The name of the Amazon Polly lexicon to
    /// delete.</param>
    ///<returns>A Boolean value indicating the success of the operation.</returns>
    public static async Task<bool> DeletePollyLexiconAsync(
        AmazonPollyClient client,
        string lexiconName)
    {
        var deleteLexiconRequest = new DeleteLexiconRequest()
        {
            Name = lexiconName,
        };

        var response = await client.DeleteLexiconAsync(deleteLexiconRequest);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

```
    }
```

- For API details, see [DeleteLexicon](#) in *AWS SDK for .NET API Reference*.

## Get a lexicon

The following code example shows how to get an Amazon Polly lexicon.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class GetLexicon
{
    public static async Task Main(string[] args)
    {
        string lexiconName = "SampleLexicon";

        var client = new AmazonPollyClient();

        await GetPollyLexiconAsync(client, lexiconName);
    }

    public static async Task GetPollyLexiconAsync(AmazonPollyClient client, string lexiconName)
    {
        var getLexiconRequest = new GetLexiconRequest()
        {
            Name = lexiconName,
        };

        try
        {
            var response = await client.GetLexiconAsync(getLexiconRequest);
            Console.WriteLine($"Lexicon:\n Name: {response.Lexicon.Name}");
            Console.WriteLine($"Content: {response.Lexicon.Content}");
        }
        catch (Exception ex)
        {
            Console.WriteLine("Error: " + ex.Message);
        }
    }
}
```

- For API details, see [GetLexicon](#) in *AWS SDK for .NET API Reference*.

## Get voices available for synthesis

The following code example shows how to get Amazon Polly voices available for synthesis.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class DescribeVoices
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();

        var allVoicesRequest = new DescribeVoicesRequest();
        var enUsVoicesRequest = new DescribeVoicesRequest()
        {
            LanguageCode = "en-US",
        };

        try
        {
            string nextToken;
            do
            {
                var allVoicesResponse = await
client.DescribeVoicesAsync(allVoicesRequest);
                nextToken = allVoicesResponse.NextToken;
                allVoicesRequest.NextToken = nextToken;

                Console.WriteLine("\nAll voices: ");
                allVoicesResponse.Voices.ForEach(voice =>
                {
                    DisplayVoiceInfo(voice);
                });
            }
            while (nextToken is not null);

            do
            {
                var enUsVoicesResponse = await
client.DescribeVoicesAsync(enUsVoicesRequest);
                nextToken = enUsVoicesResponse.NextToken;
                enUsVoicesRequest.NextToken = nextToken;

                Console.WriteLine("\nen-US voices: ");
                enUsVoicesResponse.Voices.ForEach(voice =>
                {
                    DisplayVoiceInfo(voice);
                });
            }
            while (nextToken is not null);
        }
        catch (Exception ex)
        {
            Console.WriteLine("Exception caught: " + ex.Message);
        }
    }

    public static void DisplayVoiceInfo(Voice voice)
    {
```

```
        Console.WriteLine($" Name: {voice.Name}\tGender: {voice.Gender}\tLanguageName: {voice.LanguageName}");
    }
}
```

- For API details, see [DescribeVoices](#) in *AWS SDK for .NET API Reference*.

## List pronunciation lexicons

The following code example shows how to list Amazon Polly pronunciation lexicons.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class ListLexicons
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();
        var request = new ListLexiconsRequest();

        try
        {
            Console.WriteLine("All voices: ");

            do
            {
                var response = await client.ListLexiconsAsync(request);
                request.NextToken = response.NextToken;

                response.Lexicons.ForEach(lexicon =>
                {
                    var attributes = lexicon.Attributes;
                    Console.WriteLine($"Name: {lexicon.Name}");
                    Console.WriteLine($" \tAlphabet: {attributes.Alphabet}");
                    Console.WriteLine($" \tLanguageCode: {attributes.LanguageCode}");
                    Console.WriteLine($" \tLastModified: {attributes.LastModified}");
                    Console.WriteLine($" \tLexemesCount: {attributes.LexemesCount}");
                    Console.WriteLine($" \tLexiconArn: {attributes.LexiconArn}");
                    Console.WriteLine($" \tSize: {attributes.Size}");
                });
            }
            while (request.NextToken is not null);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
    }
}
```

```
}
```

- For API details, see [ListLexicons](#) in *AWS SDK for .NET API Reference*.

## Store a pronunciation lexicon

The following code example shows how to store an Amazon Polly pronunciation lexicon.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class PutLexicon
{
    public static async Task Main()
    {
        string lexiconContent = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +
            "<lexicon version=\"1.0\" xmlns=\"http://www.w3.org/2005/01/" +
            "pronunciation-lexicon\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" " +
            "xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-lexicon" +
            "http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\" " +
            "alphabet=\"ipa\" xml:lang=\"en-US\">" +
            "<lexeme><grapheme>test1</grapheme><alias>test2</alias></lexeme>" +
            "</lexicon>";
        string lexiconName = "SampleLexicon";

        var client = new AmazonPollyClient();
        var putLexiconRequest = new PutLexiconRequest()
        {
            Name = lexiconName,
            Content = lexiconContent,
        };

        try
        {
            var response = await client.PutLexiconAsync(putLexiconRequest);
            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"Successfully created Lexicon: {lexiconName}.");
            }
            else
            {
                Console.WriteLine($"Could not create Lexicon: {lexiconName}.");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Exception caught: " + ex.Message);
        }
    }
}
```

- For API details, see [PutLexicon](#) in *AWS SDK for .NET API Reference*.

## Synthesize speech from text

The following code example shows how to synthesize speech from text with Amazon Polly.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class SynthesizeSpeech
{
    public static async Task Main()
    {
        string outputFileName = "speech.mp3";
        string text = "Twas brillig, and the slithy toves did gyre and gimbal in
the wabe";

        var client = new AmazonPollyClient();
        var response = await PollySynthesizeSpeech(client, text);

        WriteSpeechToStream(response.AudioStream, outputFileName);
    }

    ///<summary>
    /// Calls the Amazon Polly SynthesizeSpeechAsync method to convert text
    /// to speech.
    ///</summary>
    ///<param name="client">The Amazon Polly client object used to connect
    /// to the Amazon Polly service.</param>
    ///<param name="text">The text to convert to speech.</param>
    ///<returns>A SynthesizeSpeechResponse object that includes an AudioStream
    /// object with the converted text.</returns>
    private static async Task<SynthesizeSpeechResponse>
PollySynthesizeSpeech(IAmazonPolly client, string text)
    {
        var synthesizeSpeechRequest = new SynthesizeSpeechRequest()
        {
            OutputFormat = OutputFormat.Mp3,
            VoiceId = VoiceId.Joanna,
            Text = text,
        };

        var synthesizeSpeechResponse =
            await client.SynthesizeSpeechAsync(synthesizeSpeechRequest);

        return synthesizeSpeechResponse;
    }

    ///<summary>
    /// Writes the AudioStream returned from the call to
    /// SynthesizeSpeechAsync to a file in MP3 format.
    ///</summary>
    ///<param name="audioStream">The AudioStream returned from the
    /// call to the SynthesizeSpeechAsync method.</param>
```

```
    /// <param name="outputFileName">The full path to the file in which to
    /// save the audio stream.</param>
    private static void WriteSpeechToStream(Stream audioStream, string
outputFileName)
    {
        var outputStream = new FileStream(
            outputFileName,
            FileMode.Create,
            FileAccess.Write);
        byte[] buffer = new byte[2 * 1024];
        int readBytes;

        while ((readBytes = audioStream.Read(buffer, 0, 2 * 1024)) > 0)
        {
            outputStream.Write(buffer, 0, readBytes);
        }

        // Flushes the buffer to avoid losing the last second or so of
        // the synthesized text.
        outputStream.Flush();
        Console.WriteLine($"Saved {outputFileName} to disk.");
    }
}
```

Synthesize speech from text using speech marks with Amazon Polly using an AWS SDK.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;
using Amazon.Polly;
using Amazon.Polly.Model;

public class SynthesizeSpeechMarks
{
    public static async Task Main()
    {
        var client = new AmazonPollyClient();
        string outputFileName = "speechMarks.json";

        var synthesizeSpeechRequest = new SynthesizeSpeechRequest()
        {
            OutputFormat = OutputFormat.Json,
            SpeechMarkTypes = new List<string>
            {
                SpeechMarkType.Viseme,
                SpeechMarkType.Word,
            },
            VoiceId = VoiceId.Joanna,
            Text = "This is a sample text to be synthesized.",
        };

        try
        {
            using (var outputStream = new FileStream(outputFileName,
FileMode.Create, FileAccess.Write))
            {
                var synthesizeSpeechResponse = await
client.SynthesizeSpeechAsync(synthesizeSpeechRequest);
                var buffer = new byte[2 * 1024];
                int readBytes;

                var inputStream = synthesizeSpeechResponse.AudioStream;
```

```
        while ((readBytes = inputStream.Read(buffer, 0, 2 * 1024)) > 0)
        {
            outputStream.Write(buffer, 0, readBytes);
        }
    }
catch (Exception ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}
}
```

- For API details, see [SynthesizeSpeech](#) in *AWS SDK for .NET API Reference*.

## Amazon RDS examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon Relational Database Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2458\)](#)
- [Scenarios \(p. 2465\)](#)

## Actions

### Create a DB instance

The following code example shows how to create an Amazon RDS DB instance and wait for it to become available.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create an RDS DB instance with a particular set of properties. Use the action
DescribeDBInstancesAsync
/// to determine when the DB instance is ready to use.
/// </summary>
/// <param name="dBName">Name for the DB instance.</param>
/// <param name="dBInstanceIdentifier">DB instance identifier.</param>
/// <param name="parameterGroupName">DB parameter group to associate with the
instance.</param>
/// <param name="dBEngine">The engine for the DB instance.</param>
/// <param name="dBEngineVersion">Version for the DB instance.</param>
/// <param name="instanceClass">Class for the DB instance.</param>
/// <param name="allocatedStorage">The amount of storage in gibibytes (GiB) to
allocate to the DB instance.</param>
/// <param name="adminName">Admin user name.</param>
```

```
/// <param name="adminPassword">Admin user password.</param>
/// <returns>DB instance object.</returns>
public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
    string parameterGroupName, string dbEngine, string dbEngineVersion,
    string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
{
    var response = await _amazonRDS.CreateDBInstanceAsync(
        new CreateDBInstanceRequest()
    {
        DBName = dbName,
        DBInstanceIdentifier = dbInstanceIdentifier,
        DBParameterGroupName = parameterGroupName,
        Engine = dbEngine,
        EngineVersion = dbEngineVersion,
        DBInstanceClass = instanceClass,
        AllocatedStorage = allocatedStorage,
        MasterUsername = adminName,
        MasterUserPassword = adminPassword
    });

    return response.DBInstance;
}
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for .NET API Reference*.

## Create a DB parameter group

The following code example shows how to create an Amazon RDS DB parameter group.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create a new DB parameter group. Use the action DescribeDBParameterGroupsAsync
/// to determine when the DB parameter group is ready to use.
/// </summary>
/// <param name="name">Name of the DB parameter group.</param>
/// <param name="family">Family of the DB parameter group.</param>
/// <param name="description">Description of the DB parameter group.</param>
/// <returns>The new DB parameter group.</returns>
public async Task<DBParameterGroup> CreateDBParameterGroup(
    string name, string family, string description)
{
    var response = await _amazonRDS.CreateDBParameterGroupAsync(
        new CreateDBParameterGroupRequest()
    {
        DBParameterGroupName = name,
        DBParameterGroupFamily = family,
        Description = description
    });
    return response.DBParameterGroup;
}
```

- For API details, see [CreateDBParameterGroup](#) in *AWS SDK for .NET API Reference*.

## Create a snapshot of a DB instance

The following code example shows how to create a snapshot of an Amazon RDS DB instance.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create a snapshot of a DB instance.
/// </summary>
/// <param name="dbInstanceIdentifier">DB instance identifier.</param>
/// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
/// <returns>DB snapshot object.</returns>
public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier, string
snapshotIdentifier)
{
    var response = await _amazonRDS.CreateDBSnapshotAsync(
        new CreateDBSnapshotRequest()
    {
        DBSnapshotIdentifier = snapshotIdentifier,
        DBInstanceIdentifier = dbInstanceIdentifier
    });

    return response.DBSnapshot;
}
```

- For API details, see [CreateDBSnapshot](#) in *AWS SDK for .NET API Reference*.

## Delete a DB instance

The following code example shows how to delete an Amazon RDS DB instance.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Delete a particular DB instance.
/// </summary>
/// <param name="dbInstanceIdentifier">DB instance identifier.</param>
/// <returns>DB instance object.</returns>
public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
{
    var response = await _amazonRDS.DeleteDBInstanceAsync(
        new DeleteDBInstanceRequest()
    {
        DBInstanceIdentifier = dbInstanceIdentifier,
        SkipFinalSnapshot = true,
        DeleteAutomatedBackups = true
    });
}
```

```
        });

        return response.DBInstance;
    }
}
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for .NET API Reference*.

## Delete a DB parameter group

The following code example shows how to delete an Amazon RDS DB parameter group.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Delete a DB parameter group. The group cannot be a default DB parameter group
/// or be associated with any DB instances.
/// </summary>
/// <param name="name">Name of the DB parameter group.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteDBParameterGroup(string name)
{
    var response = await _amazonRDS.DeleteDBParameterGroupAsync(
        new DeleteDBParameterGroupRequest()
    {
        DBParameterGroupName = name,
    });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- For API details, see [DeleteDBParameterGroup](#) in *AWS SDK for .NET API Reference*.

## Describe DB instances

The following code example shows how to describe Amazon RDS DB instances.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Returns a list of DB instances.
/// </summary>
/// <param name="dbInstanceIdentifier">Optional name of a specific DB instance.</param>
/// <returns>List of DB instances.</returns>
public async Task<List<DBInstance>> DescribeDBInstances(string dbInstanceIdentifier
= null)
{
}
```

```
var results = new List<DBInstance>();
var instancesPaginator = _amazonRDS.Paginator.DescribeDBInstances(
    new DescribeDBInstancesRequest
    {
        DBInstanceIdentifier = dbInstanceIdentifier
    });
// Get the entire list using the paginator.
await foreach (var instances in instancesPaginator.DBInstances)
{
    results.Add(instances);
}
return results;
}
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for .NET API Reference*.

## Describe DB parameter groups

The following code example shows how to describe Amazon RDS DB parameter groups.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get descriptions of DB parameter groups.
/// </summary>
/// <param name="name">Optional name of the DB parameter group to describe.</param>
/// <returns>The list of DB parameter group descriptions.</returns>
public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string name =
null)
{
    var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
        new DescribeDBParameterGroupsRequest()
    {
        DBParameterGroupName = name
    });
    return response.DBParameterGroups;
}
```

- For API details, see [DescribeDBParameterGroups](#) in *AWS SDK for .NET API Reference*.

## Describe database engine versions

The following code example shows how to describe Amazon RDS database engine versions.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get a list of DB engine versions for a particular DB engine.
/// </summary>
/// <param name="engine">Name of the engine.</param>
/// <param name="dbParameterGroupFamily">Optional parameter group family name.</param>
/// <returns>List of DBEngineVersions.</returns>
public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string engine,
    string dbParameterGroupFamily = null)
{
    var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
        new DescribeDBEngineVersionsRequest()
    {
        Engine = engine,
        DBParameterGroupFamily = dbParameterGroupFamily
    });
    return response.DBEngineVersions;
}
```

- For API details, see [DescribeDBEngineVersions](#) in *AWS SDK for .NET API Reference*.

## Describe options for DB instances

The following code example shows how to describe options for Amazon RDS DB instances.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get a list of orderable DB instance options for a specific
/// engine and engine version.
/// </summary>
/// <param name="engine">Name of the engine.</param>
/// <param name="engineVersion">Version of the engine.</param>
/// <returns>List of OrderableDBInstanceOptions.</returns>
public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
{
    // Use a paginator to get a list of DB instance options.
    var results = new List<OrderableDBInstanceOption>();
    var paginateInstanceOptions =
_amazonRDS.Paginator.DescribeOrderableDBInstanceOptions(
    new DescribeOrderableDBInstanceOptionsRequest()
    {
        Engine = engine,
        EngineVersion = engineVersion,
    });
    // Get the entire list using the paginator.
    await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
    {
        results.Add(instanceOptions);
    }
    return results;
}
```

- For API details, see [DescribeOrderableDBInstanceOptions](#) in *AWS SDK for .NET API Reference*.

## Describe parameters in a DB parameter group

The following code example shows how to describe parameters in an Amazon RDS DB parameter group.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Get a list of DB parameters from a specific parameter group.
///</summary>
///<param name="dbParameterGroupName">Name of a specific DB parameter group.</param>
///<param name="source">Optional source for selecting parameters.</param>
///<returns>List of parameter values.</returns>
public async Task<List<Parameter>> DescribeDBParameters(string dbParameterGroupName, string source = null)
{
    var results = new List<Parameter>();
    var paginateParameters = _amazonRDS.Paginator.DescribeDBParameters(
        new DescribeDBParametersRequest()
    {
        DBParameterGroupName = dbParameterGroupName,
        Source = source
    });
    // Get the entire list using the paginator.
    await foreach (var parameters in paginateParameters.Parameters)
    {
        results.Add(parameters);
    }
    return results;
}
```

- For API details, see [DescribeDBParameters](#) in *AWS SDK for .NET API Reference*.

## Describe snapshots of DB instances

The following code example shows how to describe snapshots of Amazon RDS DB instances.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Return a list of DB snapshots for a particular DB instance.
///</summary>
///<param name="dbInstanceIdentifier">DB instance identifier.</param>
///<returns>List of DB snapshots.</returns>
```

```
public async Task<List<DBSnapshot>> DescribeDBSnapshots(string dbInstanceIdentifier)
{
    var results = new List<DBSnapshot>();
    var snapshotsPaginator = _amazonRDS.Paginator.DescribeDBSnapshots(
        new DescribeDBSnapshotsRequest()
    {
        DBInstanceIdentifier = dbInstanceIdentifier
    });

    // Get the entire list using the paginator.
    await foreach (var snapshots in snapshotsPaginator.DBSnapshots)
    {
        results.Add(snapshots);
    }
    return results;
}
```

- For API details, see [DescribeDBSnapshots](#) in *AWS SDK for .NET API Reference*.

## Update parameters in a DB parameter group

The following code example shows how to update parameters in an Amazon RDS DB parameter group.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Update a DB parameter group. Use the action DescribeDBParameterGroupsAsync
/// to determine when the DB parameter group is ready to use.
///</summary>
///<param name="name">Name of the DB parameter group.</param>
///<param name="parameters">List of parameters. Maximum of 20 per request.</param>
///<returns>The updated DB parameter group name.</returns>
public async Task<string> ModifyDBParameterGroup(
    string name, List<Parameter> parameters)
{
    var response = await _amazonRDS.ModifyDBParameterGroupAsync(
        new ModifyDBParameterGroupRequest()
    {
        DBParameterGroupName = name,
        Parameters = parameters,
    });
    return response.DBParameterGroupName;
}
```

- For API details, see [ModifyDBParameterGroup](#) in *AWS SDK for .NET API Reference*.

## Scenarios

### Get started with DB instances

The following code example shows how to:

- Create a custom DB parameter group and set parameter values.
- Create a DB instance that's configured to use the parameter group. The DB instance also contains a database.
- Take a snapshot of the instance.
- Delete the instance and parameter group.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
/// <summary>
/// Scenario for RDS DB instance example.
/// </summary>
public class RDSDInstanceScenario
{
    /*
        Before running this .NET code example, set up your development environment,
        including your credentials.

        This .NET example performs the following tasks:
        1. Returns a list of the available DB engine families using the
        DescribeDBEngineVersionsAsync method.
        2. Selects an engine family and creates a custom DB parameter group using the
        CreateDBParameterGroupAsync method.
        3. Gets the parameter groups using the DescribeDBParameterGroupsAsync method.
        4. Gets parameters in the group using the DescribeDBParameters method.
        5. Parses and displays parameters in the group.
        6. Modifies both the auto_increment_offset and auto_increment_increment parameters
        using the ModifyDBParameterGroupAsync method.
        7. Gets and displays the updated parameters using the DescribeDBParameters method
        with a source of "user".
        8. Gets a list of allowed engine versions using the DescribeDBEngineVersionsAsync
        method.
        9. Displays and selects from a list of micro instance classes available for the
        selected engine and version.
        10. Creates an RDS DB instance that contains a MySql database and uses the
        parameter group
            using the CreateDBInstanceAsync method.
        11. Waits for DB instance to be ready using the DescribeDBInstancesAsync method.
        12. Prints out the connection endpoint string for the new DB instance.
        13. Creates a snapshot of the DB instance using the CreateDBSnapshotAsync method.
        14. Waits for DB snapshot to be ready using the DescribeDBSnapshots method.
        15. Deletes the DB instance using the DeleteDBInstanceAsync method.
        16. Waits for DB instance to be deleted using the DescribeDbInstances method.
        17. Deletes the parameter group using the DeleteDBParameterGroupAsync.
    */

    private static readonly string sepBar = new('-', 80);
    private static RDSDWrapper rdsWrapper = null!;
    private static ILogger logger = null!;
    private static readonly string engine = "mysql";
    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon RDS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
```

```
.AddFilter<DebugLoggerProvider>("Microsoft", LogLevel.Information)
    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace)
.ConfigureServices(_ , services) =>
    services.AddAWSService<IAmazonRDS>()
        .AddTransient<RDSWrapper>()
)
.Build();

logger = LoggerFactory.Create(builder =>
{
    builder.AddConsole();
}).CreateLogger<RDSDInstanceScenario>();

rdsWrapper = host.Services.GetRequiredService<RDSWrapper>();

Console.WriteLine(sepBar);
Console.WriteLine(
    "Welcome to the Amazon Relational Database Service (Amazon RDS) DB instance
scenario example.");
Console.WriteLine(sepBar);

try
{
    var parameterGroupFamily = await ChooseParameterGroupFamily();

    var parameterGroup = await CreateDbParameterGroup(parameterGroupFamily);

    var parameters = await
DescribeParametersInGroup(parameterGroup.DBParameterGroupName,
    new List<string> { "auto_increment_offset",
"auto_increment_increment" });

    await ModifyParameters(parameterGroup.DBParameterGroupName, parameters);

    await DescribeUserSourceParameters(parameterGroup.DBParameterGroupName);

    var engineVersionChoice = await
ChooseDbEngineVersion(parameterGroupFamily);

    var instanceChoice = await ChooseDbInstanceClass(engine,
engineVersionChoice.EngineVersion);

    var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;

    var newInstance = await CreateRdsNewInstance(parameterGroup, engine,
engineVersionChoice.EngineVersion,
    instanceChoice.DBInstanceClass, newInstanceIdentifier);
    if (newInstance != null)
    {
        DisplayConnectionString(newInstance);

        await CreateSnapshot(newInstance);

        await DeleteRdsInstance(newInstance);
    }

    await DeleteParameterGroup(parameterGroup);

    Console.WriteLine("Scenario complete.");
    Console.WriteLine(sepBar);
}
catch (Exception ex)
{
    logger.LogError(ex, "There was a problem executing the scenario.");
}
```

```
    ///> <summary>
    ///> Choose the RDS DB parameter group family from a list of available options.
    ///> </summary>
    ///> <returns>The selected parameter group family.</returns>
    public static async Task<string> ChooseParameterGroupFamily()
    {
        Console.WriteLine(sepBar);
        // 1. Get a list of available engines.
        var engines = await rdsWrapper.DescribeDBEngineVersions(engine);

        Console.WriteLine("1. The following is a list of available DB parameter group families:");
        int i = 1;
        var parameterGroupFamilies = engines.GroupBy(e =>
e.DBParameterGroupFamily).ToList();
        foreach (var parameterGroupFamily in parameterGroupFamilies)
        {
            // List the available parameter group families.
            Console.WriteLine(
                $"\t{i}. Family: {parameterGroupFamily.Key}");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
        {
            Console.WriteLine("Select an available DB parameter group family by entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return parameterGroupFamilyChoice.Key;
    }

    ///> <summary>
    ///> Create and get information on a DB parameter group.
    ///> </summary>
    ///> <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the new DB parameter group.</param>
    ///> <returns>The new DBParameterGroup.</returns>
    public static async Task<DBParameterGroup> CreateDbParameterGroup(string dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family {dbParameterGroupFamily}:");

        var parameterGroup = await rdsWrapper.CreateDBParameterGroup(
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            dbParameterGroupFamily, "New example parameter group");

        var groupInfo =
            await rdsWrapper.DescribeDBParameterGroups(parameterGroup
                .DBParameterGroupName);

        Console.WriteLine(
            $"3. New DB parameter group: \n\t{groupInfo[0].Description}, \n\tARN {groupInfo[0].DBParameterGroupArn}");
        Console.WriteLine(sepBar);
        return parameterGroup;
    }

    ///> <summary>
```

```
    ///> Get and describe parameters from a DBParameterGroup.  
    ///></summary>  
    ///><param name="parameterGroupName">Name of the DBParameterGroup.</param>  
    ///><param name="parameterNames">Optional specific names of parameters to  
    describe.</param>  
    ///><returns>The list of requested parameters.</returns>  
    public static async Task<List<Parameter>> DescribeParametersInGroup(string  
    parameterGroupName, List<string>? parameterNames = null)  
    {  
        Console.WriteLine(sepBar);  
        Console.WriteLine("4. Get some parameters from the group.");  
        Console.WriteLine(sepBar);  
  
        var parameters =  
            await rdsWrapper.DescribeDBParameters(parameterGroupName);  
  
        var matchingParameters =  
            parameters.Where(p => parameterNames == null ||  
            parameterNames.Contains(p.ParameterName)).ToList();  
  
        Console.WriteLine("5. Parameter information:");  
        matchingParameters.ForEach(p =>  
            Console.WriteLine(  
                $"\\n\\tParameter: {p.ParameterName}." +  
                $"\\n\\tDescription: {p.Description}." +  
                $"\\n\\tAllowed Values: {p.AllowedValues}." +  
                $"\\n\\tValue: {pParameterValue}."));  
  
        Console.WriteLine(sepBar);  
  
        return matchingParameters;  
    }  
  
    ///> Modify a parameter from a DBParameterGroup.  
    ///></summary>  
    ///><param name="parameterGroupName">Name of the DBParameterGroup.</param>  
    ///><param name="parameters">The parameters to modify.</param>  
    ///><returns>Async task.</returns>  
    public static async Task ModifyParameters(string parameterGroupName,  
    List<Parameter> parameters)  
    {  
        Console.WriteLine(sepBar);  
        Console.WriteLine("6. Modify some parameters in the group.");  
  
        foreach (var p in parameters)  
        {  
            if (p.IsModifiable && p.DataType == "integer")  
            {  
                int newValue = 0;  
                while (newValue == 0)  
                {  
                    Console.WriteLine(  
                        $"Enter a new value for {p.ParameterName} from the allowed  
                        values {p.AllowedValues} ");  
  
                    var choice = Console.ReadLine();  
                    Int32.TryParse(choice, out newValue);  
                }  
  
                pParameterValue = newValue.ToString();  
            }  
        }  
  
        await rdsWrapper.ModifyDBParameterGroup(parameterGroupName, parameters);  
    }
```

```
        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string parameterGroupName)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe user source parameters in the group.");

        var parameters =
            await rdsWrapper.DescribeDBParameters(parameterGroupName, "user");

        parameters.ForEach(p =>
            Console.WriteLine(
                $"\\n\\tParameter: {p.ParameterName}." +
                $"\\n\\tDescription: {p.Description}." +
                $"\\n\\tAllowed Values: {p.AllowedValues}." +
                $"\\n\\tValue: {pParameterValue}."));
    }

    /// <summary>
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
    choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDbEngineVersion(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await rdsWrapper.DescribeDBEngineVersions(engine, dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group family
{dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\\t{i}. Engine: {version.Engine} Version {version.EngineVersion}.");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by entering a
number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }
}
```

```

    ///<summary>
    /// Choose a DB instance class for a particular engine and engine version.
    ///</summary>
    ///<param name="engine">DB engine for DB instance choice.</param>
    ///<param name="engineVersion">DB engine version for DB instance choice.</param>
    ///<returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption> ChooseDbInstanceClass(string
engine, string engineVersion)
{
    Console.WriteLine(sepBar);
    // Get a list of allowed DB instance classes.
    var allowedInstances =
        await rdsWrapper.DescribeOrderableDBInstanceOptions(engine, engineVersion);

    Console.WriteLine($"8. Available micro DB instance classes for engine {engine}
and version {engineVersion}:");
    int i = 1;

    // Filter to micro instances for this example.
    allowedInstances = allowedInstances
        .Where(i => i.DBInstanceClass.Contains("micro")).ToList();

    foreach (var instance in allowedInstances)
    {
        Console.WriteLine(
            $"{i}. Instance class: {instance.DBInstanceClass} (storage type
{instance.StorageType})");
        i++;
    }

    var choiceNumber = 0;
    while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
    {
        Console.WriteLine("9. Select an available DB instance class by entering a
number from the list above:");
        var choice = Console.ReadLine();
        Int32.TryParse(choice, out choiceNumber);
    }

    var instanceChoice = allowedInstances[choiceNumber - 1];
    Console.WriteLine(sepBar);
    return instanceChoice;
}

    ///<summary>
    /// Create a new RDS DB instance.
    ///</summary>
    ///<param name="parameterGroup">Parameter group to use for the DB instance.</
param>
    ///<param name="engineName">Engine to use for the DB instance.</param>
    ///<param name="engineVersion">Engine version to use for the DB instance.</param>
    ///<param name="instanceClass">Instance class to use for the DB instance.</param>
    ///<param name="instanceIdentifier">Instance identifier to use for the DB
instance.</param>
    ///<returns>The new DB instance.</returns>
    public static async Task<DBInstance?> CreateRdsNewInstance(DBParameterGroup
parameterGroup,
        string engineName, string engineVersion, string instanceClass, string
instanceIdentifier)
{
    Console.WriteLine(sepBar);
    Console.WriteLine($"10. Create a new DB instance with identifier
{instanceIdentifier}.");
    bool isInstanceReady = false;
    DBInstance newInstance;
    var instances = await rdsWrapper.DescribeDBInstances();

```

```
    isInstanceReady = instances.FirstOrDefault(i =>
        i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceState == "available";

    if (isInstanceReady)
    {
        Console.WriteLine("Instance already created.");
        newInstance = instances.First(i => i.DBInstanceIdentifier == instanceIdentifier);
    }
    else
    {
        Console.WriteLine("Please enter an admin user name:");
        var username = Console.ReadLine();

        Console.WriteLine("Please enter an admin password:");
        var password = Console.ReadLine();

        newInstance = await rdsWrapper.CreateDBInstance(
            "ExampleInstance",
            instanceIdentifier,
            parameterGroup.DBParameterGroupName,
            engineName,
            engineVersion,
            instanceClass,
            20,
            username,
            password
        );

        // 11. Wait for the DB instance to be ready.

        Console.WriteLine("11. Waiting for DB instance to be ready...");
        while (!isInstanceReady)
        {
            instances = await rdsWrapper.DescribeDBInstances(instanceIdentifier);
            isInstanceReady = instances.FirstOrDefault()?.DBInstanceState == "available";
            newInstance = instances.First();
            Thread.Sleep(30000);
        }
    }

    Console.WriteLine(sepBar);
    return newInstance;
}

///<summary>
/// Display a connection string for an RDS DB instance.
///</summary>
///<param name="instance">The DB instance to use to get a connection string.</param>
public static void DisplayConnectionString(DBInstance instance)
{
    Console.WriteLine(sepBar);
    // Display the connection string.
    Console.WriteLine("12. New DB instance connection string: ");
    Console.WriteLine(
        $"\\n{engine} -h {instance.Endpoint.Address} -P {instance.Endpoint.Port} " +
        $"-u {instance.MasterUsername} -p [YOUR PASSWORD]\\n");
}

Console.WriteLine(sepBar);
}

///<summary>
/// Create a snapshot from an RDS DB instance.

```

```
/// </summary>
/// <param name="instance">DB instance to use when creating a snapshot.</param>
/// <returns>The snapshot object.</returns>
public static async Task<DBSnapshot> CreateSnapshot(DBInstance instance)
{
    Console.WriteLine(sepBar);
    // Create a snapshot.
    Console.WriteLine($"13. Creating snapshot from DB instance {instance.DBInstanceIdentifier}.");
    var snapshot = await rdsWrapper.CreateDBSnapshot(instance.DBInstanceIdentifier,
"ExampleSnapshot-" + DateTime.Now.Ticks);

    // Wait for the snapshot to be available
    bool isSnapshotReady = false;

    Console.WriteLine($"14. Waiting for snapshot to be ready...");
    while (!isSnapshotReady)
    {
        var snapshots = await
rdsWrapper.DescribeDBSnapshots(instance.DBInstanceIdentifier);
        isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
        snapshot = snapshots.First();
        Thread.Sleep(30000);
    }

    Console.WriteLine(
        $"Snapshot {snapshot.DBSnapshotIdentifier} status is {snapshot.Status}.");
    Console.WriteLine(sepBar);
    return snapshot;
}

/// <summary>
/// Delete an RDS DB instance.
/// </summary>
/// <param name="instance">The DB instance to delete.</param>
/// <returns>Async task.</returns>
public static async Task DeleteRdsInstance(DBInstance newInstance)
{
    Console.WriteLine(sepBar);
    // Delete the DB instance.
    Console.WriteLine($"15. Delete the DB instance {newInstance.DBInstanceIdentifier}.");
    await rdsWrapper.DeleteDBInstance(newInstance.DBInstanceIdentifier);

    // Wait for the DB instance to delete.
    Console.WriteLine($"16. Waiting for the DB instance to delete...");
    bool isInstanceDeleted = false;

    while (!isInstanceDeleted)
    {
        var instance = await rdsWrapper.DescribeDBInstances();
        isInstanceDeleted = instance.All(i => i.DBInstanceIdentifier !=
newInstance.DBInstanceIdentifier);
        Thread.Sleep(30000);
    }

    Console.WriteLine("DB instance deleted.");
    Console.WriteLine(sepBar);
}

/// <summary>
/// Delete a DB parameter group.
/// </summary>
/// <param name="parameterGroup">The parameter group to delete.</param>
/// <returns>Async task.</returns>
public static async Task DeleteParameterGroup(DBParameterGroup parameterGroup)
```

```
{
    Console.WriteLine(sepBar);
    // Delete the parameter group.
    Console.WriteLine($"17. Delete the DB parameter group
{parameterGroup.DBParameterGroupName}.");
    await rdsWrapper.DeleteDBParameterGroup(parameterGroup.DBParameterGroupName);

    Console.WriteLine(sepBar);
}
```

Wrapper methods used by the scenario for DB instance actions.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with DB
instance operations.
/// </summary>
public partial class RDSWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public RDSWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }

    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="dbParameterGroupFamily">Optional parameter group family name.</param>
    /// <returns>List of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>> DescribeDBEngineVersions(string engine,
        string dbParameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = dbParameterGroupFamily
            });
        return response.DBEngineVersions;
    }

    /// <summary>
    /// Get a list of orderable DB instance options for a specific
    /// engine and engine version.
    /// </summary>
    /// <param name="engine">Name of the engine.</param>
    /// <param name="engineVersion">Version of the engine.</param>
    /// <returns>List of OrderableDBInstanceOptions.</returns>
    public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptions(string engine, string engineVersion)
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginator.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
```

```

        EngineVersion = engineVersion,
    });
    // Get the entire list using the paginator.
    await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceState)
    {
        results.Add(instanceOptions);
    }
    return results;
}

Returns a list of DB instances.
Optional name of a specific DB instance.</param>
List of DB instances.</returns>
public async Task<List<DBInstance>> DescribeDBInstances(string dbInstanceIdentifier
= null)
{
    var results = new List<DBInstance>();
    var instancesPaginator = _amazonRDS.Paginator.DescribeDBInstances(
        new DescribeDBInstancesRequest
    {
        DBInstanceIdentifier = dbInstanceIdentifier
    });
    // Get the entire list using the paginator.
    await foreach (var instances in instancesPaginator.DBInstances)
    {
        results.Add(instances);
    }
    return results;
}

Create an RDS DB instance with a particular set of properties. Use the action DescribeDBInstancesAsync
Name for the DB instance.</param>
DB instance identifier.</param>
DB parameter group to associate with the instance.</param>
The engine for the DB instance.</param>
Version for the DB instance.</param>
Class for the DB instance.</param>
The amount of storage in gibibytes (GiB) to allocate to the DB instance.</param>
Admin user name.</param>
Admin user password.</param>
DB instance object.</returns>
public async Task<DBInstance> CreateDBInstance(string dbName, string
dbInstanceIdentifier,
        string parameterGroupName, string dbEngine, string dbEngineVersion,
        string instanceClass, int allocatedStorage, string adminName, string
adminPassword)
{
    var response = await _amazonRDS.CreateDBInstanceAsync(
        new CreateDBInstanceRequest()
    {
        DBName = dbName,
        DBInstanceIdentifier = dbInstanceIdentifier,

```

```

        DBParameterGroupName = parameterGroupName,
        Engine = dbEngine,
        EngineVersion = dbEngineVersion,
        DBInstanceClass = instanceClass,
        AllocatedStorage = allocatedStorage,
        MasterUsername = adminName,
        MasterUserPassword = adminPassword
    });

    return response.DBInstance;
}

//// <summary>
//// Delete a particular DB instance.
//// </summary>
//// <param name="dbInstanceIdentifier">DB instance identifier.</param>
//// <returns>DB instance object.</returns>
public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
{
    var response = await _amazonRDS.DeleteDBInstanceAsync(
        new DeleteDBInstanceRequest()
    {
        DBInstanceIdentifier = dbInstanceIdentifier,
        SkipFinalSnapshot = true,
        DeleteAutomatedBackups = true
    });

    return response.DBInstance;
}

```

Wrapper methods used by the scenario for DB parameter groups.

```

/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
/// parameter groups.
/// </summary>
public partial class RDSServerless
{

    /// <summary>
    /// Get descriptions of DB parameter groups.
    /// </summary>
    /// <param name="name">Optional name of the DB parameter group to describe.</param>
    /// <returns>The list of DB parameter group descriptions.</returns>
    public async Task<List<DBParameterGroup>> DescribeDBParameterGroups(string name =
null)
    {
        var response = await _amazonRDS.DescribeDBParameterGroupsAsync(
            new DescribeDBParameterGroupsRequest()
        {
            DBParameterGroupName = name
        });
        return response.DBParameterGroups;
    }

    /// <summary>
    /// Create a new DB parameter group. Use the action DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    /// </summary>
    /// <param name="parameterGroupDescription">DB parameter group description to
    /// create. This is required.
    /// </param>
    /// <param name="engineVersion">DB engine version to use. This is required.
    /// </param>
    /// <param name="allocatedStorage">Allocated storage for the DB instance. This is
    /// required.
    /// </param>
    /// <param name="instanceClass">DB instance class to use. This is required.
    /// </param>
    /// <param name="masterUsername">Master user name for the DB instance. This is
    /// required.
    /// </param>
    /// <param name="masterUserPassword">Master user password for the DB instance. This
    /// is required.
    /// </param>
    /// <param name="skipFinalSnapshot">A value indicating whether to skip the final
    /// snapshot of the DB instance. If true, the DB instance is deleted immediately.
    /// Otherwise, the final snapshot is taken before the DB instance is deleted.
    /// </param>
    /// <param name="deleteAutomatedBackups">A value indicating whether to delete
    /// automated backups for the DB instance. If true, automated backups are deleted
    /// immediately. Otherwise, they are retained.
    /// </param>
    /// <param name="dbInstanceIdentifier">The identifier for the DB instance to
    /// delete. This is required.
    /// </param>
    public async Task<DBInstance> DeleteDBInstance(string dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
        {
            DBInstanceIdentifier = dbInstanceIdentifier
        });

        return response.DBInstance;
    }
}

```

```

    ///</summary>
    ///<param name="name">Name of the DB parameter group.</param>
    ///<param name="family">Family of the DB parameter group.</param>
    ///<param name="description">Description of the DB parameter group.</param>
    ///<returns>The new DB parameter group.</returns>
    public async Task<DBParameterGroup> CreateDBParameterGroup(
        string name, string family, string description)
    {
        var response = await _amazonRDS.CreateDBParameterGroupAsync(
            new CreateDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                DBParameterGroupFamily = family,
                Description = description
            });
        return response.DBParameterGroup;
    }

    ///<summary>
    /// Update a DB parameter group. Use the action DescribeDBParameterGroupsAsync
    /// to determine when the DB parameter group is ready to use.
    ///</summary>
    ///<param name="name">Name of the DB parameter group.</param>
    ///<param name="parameters">List of parameters. Maximum of 20 per request.</param>
    ///<returns>The updated DB parameter group name.</returns>
    public async Task<string> ModifyDBParameterGroup(
        string name, List<Parameter> parameters)
    {
        var response = await _amazonRDS.ModifyDBParameterGroupAsync(
            new ModifyDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
                Parameters = parameters,
            });
        return response.DBParameterGroupName;
    }

    ///<summary>
    /// Delete a DB parameter group. The group cannot be a default DB parameter group
    /// or be associated with any DB instances.
    ///</summary>
    ///<param name="name">Name of the DB parameter group.</param>
    ///<returns>True if successful.</returns>
    public async Task<bool> DeleteDBParameterGroup(string name)
    {
        var response = await _amazonRDS.DeleteDBParameterGroupAsync(
            new DeleteDBParameterGroupRequest()
            {
                DBParameterGroupName = name,
            });
        return response.StatusCode == HttpStatusCode.OK;
    }

    ///<summary>
    /// Get a list of DB parameters from a specific parameter group.
    ///</summary>
    ///<param name="dbParameterGroupName">Name of a specific DB parameter group.</param>
    ///<param name="source">Optional source for selecting parameters.</param>
    ///<returns>List of parameter values.</returns>

```

```
public async Task<List<Parameter>> DescribeDBParameters(string dbParameterGroupName, string source = null)
{
    var results = new List<Parameter>();
    var paginateParameters = _amazonRDS.Paginator.DescribeDBParameters(
        new DescribeDBParametersRequest()
    {
        DBParameterGroupName = dbParameterGroupName,
        Source = source
    });
    // Get the entire list using the paginator.
    await foreach (var parameters in paginateParameters.Parameters)
    {
        results.Add(parameters);
    }
    return results;
}
```

Wrapper methods used by the scenario for DB snapshot actions.

```
/// <summary>
/// Wrapper methods to use Amazon Relational Database Service (Amazon RDS) with
/// snapshots.
/// </summary>
public partial class RDSService
{

    /// <summary>
    /// Create a snapshot of a DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBSnapshot> CreateDBSnapshot(string dbInstanceIdentifier, string snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBSnapshotAsync(
            new CreateDBSnapshotRequest()
        {
            DBSnapshotIdentifier = snapshotIdentifier,
            DBInstanceIdentifier = dbInstanceIdentifier
        });

        return response.DBSnapshot;
    }

    /// <summary>
    /// Return a list of DB snapshots for a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBSnapshot>> DescribeDBSnapshots(string dbInstanceIdentifier)
    {
        var results = new List<DBSnapshot>();
        var snapshotsPaginator = _amazonRDS.Paginator.DescribeDBSnapshots(
            new DescribeDBSnapshotsRequest()
        {
            DBInstanceIdentifier = dbInstanceIdentifier
        });
    }
```

```
// Get the entire list using the paginator.  
await foreach (var snapshots in snapshotsPaginator.DBSnapshots)  
{  
    results.Add(snapshots);  
}  
return results;  
}
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [CreateDBInstance](#)
  - [CreateDBParameterGroup](#)
  - [CreateDBSnapshot](#)
  - [DeleteDBInstance](#)
  - [DeleteDBParameterGroup](#)
  - [DescribeDBEngineVersions](#)
  - [DescribeDBInstances](#)
  - [DescribeDBParameterGroups](#)
  - [DescribeDBParameters](#)
  - [DescribeDBSnapshots](#)
  - [DescribeOrderableDBInstanceOptions](#)
  - [ModifyDBParameterGroup](#)

## Amazon Rekognition examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon Rekognition.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2479\)](#)

## Actions

### Compare faces in an image against a reference image

The following code example shows how to compare faces in an image against a reference image with Amazon Rekognition.

For more information, see [Comparing faces in images](#).

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.IO;
```

```
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

///<summary>
/// Uses the Amazon Rekognition Service to compare faces in two images.
/// The example uses the AWS SDK for .NET 3.7 and .NET Core 5.0.
///</summary>
public class CompareFaces
{
    public static async Task Main()
    {
        float similarityThreshold = 70F;
        string sourceImage = "source.jpg";
        string targetImage = "target.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        Amazon.Rekognition.Model.Image imageSource = new
        Amazon.Rekognition.Model.Image();

        try
        {
            using FileStream fs = new FileStream(sourceImage, FileMode.Open,
FileAccess.Read);
            byte[] data = new byte[fs.Length];
            fs.Read(data, 0, (int)fs.Length);
            imageSource.Bytes = new MemoryStream(data);
        }
        catch (Exception)
        {
            Console.WriteLine($"Failed to load source image: {sourceImage}");
            return;
        }

        Amazon.Rekognition.Model.Image imageTarget = new
        Amazon.Rekognition.Model.Image();

        try
        {
            using FileStream fs = new FileStream(targetImage, FileMode.Open,
FileAccess.Read);
            byte[] data = new byte[fs.Length];
            data = new byte[fs.Length];
            fs.Read(data, 0, (int)fs.Length);
            imageTarget.Bytes = new MemoryStream(data);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Failed to load target image: {targetImage}");
            Console.WriteLine(ex.Message);
            return;
        }

        var compareFacesRequest = new CompareFacesRequest
        {
            SourceImage = imageSource,
            TargetImage = imageTarget,
            SimilarityThreshold = similarityThreshold,
        };

        // Call operation
        var compareFacesResponse = await
rekognitionClient.CompareFacesAsync(compareFacesRequest);

        // Display results
```

```
compareFacesResponse.FaceMatches.ForEach(match =>
{
    ComparedFace face = match.Face;
    BoundingBox position = face.BoundingBox;
    Console.WriteLine($"Face at {position.Left} {position.Top} matches with
{match.Similarity}% confidence.");
});

Console.WriteLine($"Found {compareFacesResponse.UnmatchedFaces.Count}
face(s) that did not match.");
}
```

- For API details, see [CompareFaces](#) in *AWS SDK for .NET API Reference*.

## Create a collection

The following code example shows how to create an Amazon Rekognition collection.

For more information, see [Creating a collection](#).

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses Amazon Rekognition to create a collection to which you can add
/// faces using the IndexFaces operation. The example was created using
/// the AWS SDK for .NET 3.7 and .NET Core 5.0.
/// </summary>
public class CreateCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine("Creating collection: " + collectionId);

        var createCollectionRequest = new CreateCollectionRequest
        {
            CollectionId = collectionId,
        };

        CreateCollectionResponse createCollectionResponse = await
rekognitionClient.CreateCollectionAsync(createCollectionRequest);
        Console.WriteLine($"CollectionArn :
{createCollectionResponse.CollectionArn}");
        Console.WriteLine($"Status code : {createCollectionResponse.StatusCode}");
    }
}
```

- For API details, see [CreateCollection](#) in *AWS SDK for .NET API Reference*.

## Delete a collection

The following code example shows how to delete an Amazon Rekognition collection.

For more information, see [Deleting a collection](#).

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

///<summary>
/// Uses the Amazon Rekognition Service to delete an existing collection.
/// The example was created using the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
///</summary>
public class DeleteCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine("Deleting collection: " + collectionId);

        var deleteCollectionRequest = new DeleteCollectionRequest()
        {
            CollectionId = collectionId,
        };

        var deleteCollectionResponse = await
rekognitionClient.DeleteCollectionAsync(deleteCollectionRequest);
        Console.WriteLine($"{collectionId}:
{deleteCollectionResponse.StatusCode}");
    }
}
```

- For API details, see [DeleteCollection in AWS SDK for .NET API Reference](#).

## Delete faces from a collection

The following code example shows how to delete faces from an Amazon Rekognition collection.

For more information, see [Deleting faces from a collection](#).

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
```

```
using Amazon.Rekognition.Model;

<:/// <summary>
<:/// Uses the Amazon Rekognition Service to delete one or more faces from
<:/// a Rekognition collection. This example was created using the AWS SDK
<:/// for .NET version 3.7 and .NET Core 5.0.
<:/// </summary>
public class DeleteFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        var faces = new List<string> { "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" };

        var rekognitionClient = new AmazonRekognitionClient();

        var deleteFacesRequest = new DeleteFacesRequest()
        {
            CollectionId = collectionId,
            FaceIds = faces,
        };

        DeleteFacesResponse deleteFacesResponse = await
rekognitionClient.DeleteFacesAsync(deleteFacesRequest);
        deleteFacesResponse.DeletedFaces.ForEach(face =>
        {
            Console.WriteLine($"FaceID: {face}");
        });
    }
}
```

- For API details, see [DeleteFaces](#) in *AWS SDK for .NET API Reference*.

## Describe a collection

The following code example shows how to describe an Amazon Rekognition collection.

For more information, see [Describing a collection](#).

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

<:/// <summary>
<:/// Uses the Amazon Rekognition Service to describe the contents of a
<:/// collection. The example was created using the AWS SDK for .NET version
<:/// 3.7 and .NET Core 5.0.
<:/// </summary>
public class DescribeCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();
```

```
        string collectionId = "MyCollection";
        Console.WriteLine($"Describing collection: {collectionId}");

        var describeCollectionRequest = new DescribeCollectionRequest()
        {
            CollectionId = collectionId,
        };

        var describeCollectionResponse = await
rekognitionClient.DescribeCollectionAsync(describeCollectionRequest);
        Console.WriteLine($"Collection ARN:
{describeCollectionResponse.CollectionARN}");
        Console.WriteLine($"Face count: {describeCollectionResponse.FaceCount}");
        Console.WriteLine($"Face model version:
{describeCollectionResponse.FaceModelVersion}");
        Console.WriteLine($"Created:
{describeCollectionResponse.CreationTimestamp}");
    }
}
```

- For API details, see [DescribeCollection](#) in *AWS SDK for .NET API Reference*.

## Detect faces in an image

The following code example shows how to detect faces in an image with Amazon Rekognition.

For more information, see [Detecting faces in an image](#).

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket. This
/// example uses the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DetectFaces
{
    public static async Task Main()
    {
        string photo = "input.jpg";
        string bucket = "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectFacesRequest = new DetectFacesRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,

```

```

        },
    },

    // Attributes can be "ALL" or "DEFAULT".
    // "DEFAULT": BoundingBox, Confidence, Landmarks, Pose, and Quality.
    // "ALL": See https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/
Rekognition/TFaceDetail.html
    Attributes = new List<string>() { "ALL" },
};

try
{
    DetectFacesResponse detectFacesResponse = await
rekognitionClient.DetectFacesAsync(detectFacesRequest);
    bool hasAll = detectFacesRequest.Attributes.Contains("ALL");
    foreach (FaceDetail face in detectFacesResponse.FaceDetails)
    {
        Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
left={face.BoundingBox.Top} width={face.BoundingBox.Width}
height={face.BoundingBox.Height}");
        Console.WriteLine($"Confidence: {face.Confidence}");
        Console.WriteLine($"Landmarks: {face.Landmarks.Count}");
        Console.WriteLine($"Pose: pitch={face.Pose.Pitch}
roll={face.Pose.Roll} yaw={face.Pose.Yaw}");
        Console.WriteLine($"Brightness:
{face.Quality.Brightness}\tSharpness: {face.Quality.Sharpness}");

        if (hasAll)
        {
            Console.WriteLine($"Estimated age is between
{face.AgeRange.Low} and {face.AgeRange.High} years old.");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
}

```

Display bounding box information for all faces in an image.

```

using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to display the details of the
/// bounding boxes around the faces detected in an image. This example was
/// created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class ImageOrientationBoundingBox
{
    public static async Task Main()
    {
        string photo = @"D:\Development\AWS-Examples\Rekognition\target.jpg"; // 
"photo.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

```

```
var image = new Amazon.Rekognition.Model.Image();
try
{
    using var fs = new FileStream(photo, FileMode.Open, FileAccess.Read);
    byte[] data = null;
    data = new byte[fs.Length];
    fs.Read(data, 0, (int)fs.Length);
    image.Bytes = new MemoryStream(data);
}
catch (Exception)
{
    Console.WriteLine("Failed to load file " + photo);
    return;
}

int height;
int width;

// Used to extract original photo width/height
using (var imageBitmap = new Bitmap(photo))
{
    height = imageBitmap.Height;
    width = imageBitmap.Width;
}

Console.WriteLine("Image Information:");
Console.WriteLine(photo);
Console.WriteLine("Image Height: " + height);
Console.WriteLine("Image Width: " + width);

try
{
    var detectFacesRequest = new DetectFacesRequest()
    {
        Image = image,
        Attributes = new List<string>() { "ALL" },
    };

    DetectFacesResponse detectFacesResponse = await
rekognitionClient.DetectFacesAsync(detectFacesRequest);
    detectFacesResponse.FaceDetails.ForEach(face =>
    {
        Console.WriteLine("Face:");
        ShowBoundingBoxPositions(
            height,
            width,
            face.BoundingBox,
            detectFacesResponse.OrientationCorrection);

        Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}\
left={face.BoundingBox.Top} width={face.BoundingBox.Width}\
height={face.BoundingBox.Height}");
        Console.WriteLine($"The detected face is estimated to be between\
{face.AgeRange.Low} and {face.AgeRange.High} years old.\n");
    });
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
}

/// <summary>
/// Display the bounding box information for an image.
/// </summary>
```

```
    ///<param name="imageHeight">The height of the image.</param>
    ///<param name="imageWidth">The width of the image.</param>
    ///<param name="box">The bounding box for a face found within the image.</param>
    public static void ShowBoundingBoxPositions(int imageHeight, int imageWidth,
        BoundingBox box, string rotation)
    {
        float left;
        float top;

        if (rotation == null)
        {
            Console.WriteLine("No estimated orientation. Check Exif data.");
            return;
        }

        // Calculate face position based on image orientation.
        switch (rotation)
        {
            case "ROTATE_0":
                left = imageWidth * box.Left;
                top = imageHeight * box.Top;
                break;
            case "ROTATE_90":
                left = imageHeight * (1 - (box.Top + box.Height));
                top = imageWidth * box.Left;
                break;
            case "ROTATE_180":
                left = imageWidth - (imageWidth * (box.Left + box.Width));
                top = imageHeight * (1 - (box.Top + box.Height));
                break;
            case "ROTATE_270":
                left = imageHeight * box.Top;
                top = imageWidth * (1 - box.Left - box.Width);
                break;
            default:
                Console.WriteLine("No estimated orientation information. Check Exif
data.");
                return;
        }

        // Display face location information.
        Console.WriteLine($"Left: {left}");
        Console.WriteLine($"Top: {top}");
        Console.WriteLine($"Face Width: {imageWidth * box.Width}");
        Console.WriteLine($"Face Height: {imageHeight * box.Height}");
    }
}
```

- For API details, see [DetectFaces](#) in *AWS SDK for .NET API Reference*.

## Detect labels in an image

The following code example shows how to detect labels in an image with Amazon Rekognition.

For more information, see [Detecting labels in an image](#).

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect labels within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket. This
/// example was created using the AWS SDK for .NET and .NET Core 5.0.
/// </summary>
public class DetectLabels
{
    public static async Task Main()
    {
        string photo = "del_river_02092020_01.jpg"; // "input.jpg";
        string bucket = "igsmiths3photos"; // "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectlabelsRequest = new DetectLabelsRequest
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
                MaxLabels = 10,
                MinConfidence = 75F,
            };
            try
            {
                DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (Label label in detectLabelsResponse.Labels)
                {
                    Console.WriteLine($"Name: {label.Name} Confidence:
{label.Confidence}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
}
```

Detect labels in an image file stored on your computer.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect labels within an image
/// stored locally. This example was created using the AWS SDK for .NET
```

```
/// and .NET Core 5.0.  
/// </summary>  
public class DetectLabelsLocalFile  
{  
    public static async Task Main()  
    {  
        string photo = "input.jpg";  
  
        var image = new Amazon.Rekognition.Model.Image();  
        try  
        {  
            using var fs = new FileStream(photo, FileMode.Open, FileAccess.Read);  
            byte[] data = null;  
            data = new byte[fs.Length];  
            fs.Read(data, 0, (int)fs.Length);  
            image.Bytes = new MemoryStream(data);  
        }  
        catch (Exception)  
        {  
            Console.WriteLine("Failed to load file " + photo);  
            return;  
        }  
  
        var rekognitionClient = new AmazonRekognitionClient();  
  
        var detectlabelsRequest = new DetectLabelsRequest  
        {  
            Image = image,  
            MaxLabels = 10,  
            MinConfidence = 77F,  
        };  
  
        try  
        {  
            DetectLabelsResponse detectLabelsResponse = await  
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);  
            Console.WriteLine($"Detected labels for {photo}");  
            foreach (Label label in detectLabelsResponse.Labels)  
            {  
                Console.WriteLine($"{label.Name}: {label.Confidence}");  
            }  
        }  
        catch (Exception ex)  
        {  
            Console.WriteLine(ex.Message);  
        }  
    }  
}
```

- For API details, see [DetectLabels](#) in *AWS SDK for .NET API Reference*.

## Detect moderation labels in an image

The following code example shows how to detect moderation labels in an image with Amazon Rekognition. Moderation labels identify content that may be inappropriate for some audiences.

For more information, see [Detecting inappropriate images](#).

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect unsafe content in a
/// JPEG or PNG format image. This example was created using the AWS SDK
/// for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class DetectModerationLabels
{
    public static async Task Main(string[] args)
    {
        string photo = "input.jpg";
        string bucket = "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectModerationLabelsRequest = new DetectModerationLabelsRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },
            MinConfidence = 60F,
        };

        try
        {
            var detectModerationLabelsResponse = await
rekognitionClient.DetectModerationLabelsAsync(detectModerationLabelsRequest);
            Console.WriteLine("Detected labels for " + photo);
            foreach (ModerationLabel label in
detectModerationLabelsResponse.ModerationLabels)
            {
                Console.WriteLine($"Label: {label.Name}");
                Console.WriteLine($"Confidence: {label.Confidence}");
                Console.WriteLine($"Parent: {label.ParentName}");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}
```

- For API details, see [DetectModerationLabels](#) in *AWS SDK for .NET API Reference*.

## Detect text in an image

The following code example shows how to detect text in an image with Amazon Rekognition.

For more information, see [Detecting text in an image](#).

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect text in an image. The
/// example was created using the AWS SDK for .NET version 3.7 and .NET
/// Core 5.0.
/// </summary>
public class DetectText
{
    public static async Task Main()
    {
        string photo = "Dad_photographer.jpg"; // "input.jpg";
        string bucket = "igsmiths3photos"; // "bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectTextRequest = new DetectTextRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },
        };

        try
        {
            DetectTextResponse detectTextResponse = await
rekognitionClient.DetectTextAsync(detectTextRequest);
            Console.WriteLine($"Detected lines and words for {photo}");
            detectTextResponse.TextDetections.ForEach(text =>
            {
                Console.WriteLine($"Detected: {text.DetectedText}");
                Console.WriteLine($"Confidence: {text.Confidence}");
                Console.WriteLine($"Id : {text.Id}");
                Console.WriteLine($"Parent Id: {text.ParentId}");
                Console.WriteLine($"Type: {text.Type}");
            });
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```

- For API details, see [DetectText](#) in *AWS SDK for .NET API Reference*.

## Get information about celebrities

The following code example shows how to get information about celebrities using Amazon Rekognition.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to retrieve information about the
/// celebrity identified by the supplied celebrity Id. This example was
/// created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class CelebrityInfo
{
    public static async Task Main()
    {
        string celebId = "nnnnnnnn";

        var rekognitionClient = new AmazonRekognitionClient();

        var celebrityInfoRequest = new GetCelebrityInfoRequest
        {
            Id = celebId,
        };

        Console.WriteLine($"Getting information for celebrity: {celebId}");

        var celebrityInfoResponse = await
rekognitionClient.GetCelebrityInfoAsync(celebrityInfoRequest);

        // Display celebrity information.
        Console.WriteLine($"celebrity name: {celebrityInfoResponse.Name}");
        Console.WriteLine("Further information (if available):");
        celebrityInfoResponseUrls.ForEach(url =>
        {
            Console.WriteLine(url);
        });
    }
}
```

- For API details, see [GetCelebrityInfo in AWS SDK for .NET API Reference](#).

## Index faces to a collection

The following code example shows how to index faces in an image and add them to an Amazon Rekognition collection.

For more information, see [Adding faces to a collection](#).

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces in an image
/// that has been uploaded to an Amazon Simple Storage Service (Amazon S3)
/// bucket and then adds the information to a collection. The example was
/// created using the AWS SDK for .NET and .NET Core 5.0.
/// </summary>
public class AddFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection2";
        string bucket = "doc-example-bucket";
        string photo = "input.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        var image = new Image
        {
            S3Object = new S3Object
            {
                Bucket = bucket,
                Name = photo,
            },
        };

        var indexFacesRequest = new IndexFacesRequest
        {
            Image = image,
            CollectionId = collectionId,
            ExternalImageId = photo,
            DetectionAttributes = new List<string>() { "ALL" },
        };

        IndexFacesResponse indexFacesResponse = await
rekognitionClient.IndexFacesAsync(indexFacesRequest);

        Console.WriteLine($"{photo} added");
        foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
        {
            Console.WriteLine($"Face detected: Faceid is
{faceRecord.Face.FaceId}");
        }
    }
}
```

- For API details, see [IndexFaces](#) in *AWS SDK for .NET API Reference*.

## List collections

The following code example shows how to list Amazon Rekognition collections.

For more information, see [Listing collections](#).

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to list the collection IDs in the
/// default user account. This example was created using the AWS SDK for
/// .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class ListCollections
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        Console.WriteLine("Listing collections");
        int limit = 10;

        var listCollectionsRequest = new ListCollectionsRequest
        {
            MaxResults = limit,
        };

        var listCollectionsResponse = new ListCollectionsResponse();

        do
        {
            if (listCollectionsResponse is not null)
            {
                listCollectionsRequest.NextToken =
listCollectionsResponse.NextToken;
            }

            listCollectionsResponse = await
rekognitionClient.ListCollectionsAsync(listCollectionsRequest);

            listCollectionsResponse.CollectionIds.ForEach(id =>
            {
                Console.WriteLine(id);
            });
        }
        while (listCollectionsResponse.NextToken is not null);
    }
}
```

- For API details, see [ListCollections](#) in *AWS SDK for .NET API Reference*.

## List faces in a collection

The following code example shows how to list faces in an Amazon Rekognition collection.

For more information, see [Listing faces in a collection](#).

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to retrieve the list of faces
/// stored in a collection. The example was created using AWS SDK for
/// .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class ListFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection2";

        var rekognitionClient = new AmazonRekognitionClient();

        var listFacesResponse = new ListFacesResponse();
        Console.WriteLine($"Faces in collection {collectionId}");

        var listFacesRequest = new ListFacesRequest
        {
            CollectionId = collectionId,
            MaxResults = 1,
        };

        do
        {
            listFacesResponse = await
rekognitionClient.ListFacesAsync(listFacesRequest);
            listFacesResponse.Faces.ForEach(face =>
            {
                Console.WriteLine(face.FaceId);
            });

            listFacesRequest.NextToken = listFacesResponse.NextToken;
        }
        while (!string.IsNullOrEmpty(listFacesResponse.NextToken));
    }
}
```

- For API details, see [ListFaces](#) in *AWS SDK for .NET API Reference*.

## Recognize celebrities in an image

The following code example shows how to recognize celebrities in an image with Amazon Rekognition.

For more information, see [Recognizing celebrities in an image](#).

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to identify celebrities in a photo.
/// This example was created using the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
/// </summary>
public class CelebritiesInImage
{
    public static async Task Main(string[] args)
    {
        string photo = "moviestars.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        var recognizeCelebritiesRequest = new RecognizeCelebritiesRequest();

        var img = new Amazon.Rekognition.Model.Image();
        byte[] data = null;
        try
        {
            using var fs = new FileStream(photo, FileMode.Open, FileAccess.Read);
            data = new byte[fs.Length];
            fs.Read(data, 0, (int)fs.Length);
        }
        catch (Exception)
        {
            Console.WriteLine($"Failed to load file {photo}");
            return;
        }

        img.Bytes = new MemoryStream(data);
        recognizeCelebritiesRequest.Image = img;

        Console.WriteLine($"Looking for celebrities in image {photo}\n");

        var recognizeCelebritiesResponse = await
rekognitionClient.RecognizeCelebritiesAsync(recognizeCelebritiesRequest);

        Console.WriteLine($"{recognizeCelebritiesResponse.CelebrityFaces.Count} celebrity(s) were recognized.\n");
        recognizeCelebritiesResponse.CelebrityFaces.ForEach(celeb =>
        {
            Console.WriteLine($"Celebrity recognized: {celeb.Name}");
            Console.WriteLine($"Celebrity ID: {celeb.Id}");
            BoundingBox boundingBox = celeb.Face.BoundingBox;
            Console.WriteLine($"position: {boundingBox.Left} {boundingBox.Top}");
            Console.WriteLine("Further information (if available):");
            celebUrls.ForEach(url =>
            {
                Console.WriteLine(url);
            });
        });

        Console.WriteLine($"{recognizeCelebritiesResponse.UnrecognizedFaces.Count} face(s) were unrecognized.");
    }
}
```

- For API details, see [RecognizeCelebrities](#) in *AWS SDK for .NET API Reference*.

## Search for faces in a collection

The following code example shows how to search for faces in an Amazon Rekognition collection that match another face from the collection.

For more information, see [Searching for a face \(face ID\)](#).

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to find faces in an image that
/// match the face Id provided in the method request. This example was
/// created using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class SearchFacesMatchingId
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        string faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";

        var rekognitionClient = new AmazonRekognitionClient();

        // Search collection for faces matching the face id.
        var searchFacesRequest = new SearchFacesRequest
        {
            CollectionId = collectionId,
            FaceId = faceId,
            FaceMatchThreshold = 70F,
            MaxFaces = 2,
        };

        SearchFacesResponse searchFacesResponse = await
rekognitionClient.SearchFacesAsync(searchFacesRequest);

        Console.WriteLine("Face matching faceId " + faceId);

        Console.WriteLine("Matche(s): ");
        searchFacesResponse.FaceMatches.ForEach(face =>
        {
            Console.WriteLine($"FaceId: {face.Face.FaceId} Similarity:
{face.Similarity}");
        });
    }
}
```

- For API details, see [SearchFaces](#) in *AWS SDK for .NET API Reference*.

## Search for faces in a collection compared to a reference image

The following code example shows how to search for faces in an Amazon Rekognition collection compared to a reference image.

For more information, see [Searching for a face \(image\)](#).

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

///<summary>
/// Uses the Amazon Rekognition Service to search for images matching those
/// in a collection. The example was created using the AWS SDK for .NET
/// version 3.7 and .NET Core 5.0.
///</summary>
public class SearchFacesMatchingImage
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        string bucket = "bucket";
        string photo = "input.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        // Get an image object from S3 bucket.
        var image = new Image()
        {
            S3Object = new S3Object()
            {
                Bucket = bucket,
                Name = photo,
            },
        };

        var searchFacesByImageRequest = new SearchFacesByImageRequest()
        {
            CollectionId = collectionId,
            Image = image,
            FaceMatchThreshold = 70F,
            MaxFaces = 2,
        };

        SearchFacesByImageResponse searchFacesByImageResponse = await
rekognitionClient.SearchFacesByImageAsync(searchFacesByImageRequest);

        Console.WriteLine("Faces matching largest face in image from " + photo);
        searchFacesByImageResponse.FaceMatches.ForEach(face =>
        {
            Console.WriteLine($"FaceId: {face.Face.FaceId}, Similarity:
{face.Similarity}");
        });
    }
}
```

- For API details, see [SearchFacesByImage](#) in *AWS SDK for .NET API Reference*.

## Route 53 domain registration examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon Route 53 domain registration.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Get started

#### Hello Route 53 domain registration

The following code example shows how to get started using Amazon Route 53 domain registration.

.NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static class HelloRoute53Domains
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
        // the Amazon Route 53 domain registration service.
        // Use your AWS profile name, or leave it blank to use the default profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRoute53Domains>()
            ).Build();

        // Now the client is available for injection.
        var route53Client =
            host.Services.GetRequiredService<IAmazonRoute53Domains>();

        // You can use await and any of the async methods to get a response.
        var response = await route53Client.ListPricesAsync(new ListPricesRequest
        { Tld = "com" });
        Console.WriteLine($"Hello Amazon Route 53 Domains! Following are prices
        for .com domain operations:");
        var comPrices = response.Prices.FirstOrDefault();
        if (comPrices != null)
        {
            Console.WriteLine($"\\tRegistration:
{comPrices.RegistrationPrice?.Price} {comPrices.RegistrationPrice?.Currency}");
            Console.WriteLine($"\\tRenewal: {comPrices.RenewalPrice?.Price}
{comPrices.RenewalPrice?.Currency}");
        }
    }
}
```

- For API details, see [ListPrices](#) in *AWS SDK for .NET API Reference*.

## Topics

- [Actions \(p. 2500\)](#)
- [Scenarios \(p. 2506\)](#)

## Actions

### Check domain availability

The following code example shows how to check the availability of a domain.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Check the availability of a domain name.
/// </summary>
/// <param name="domain">The domain to check for availability.</param>
/// <returns>An availability result string.</returns>
public async Task<string> CheckDomainAvailability(string domain)
{
    var result = await _amazonRoute53Domains.CheckDomainAvailabilityAsync(
        new CheckDomainAvailabilityRequest
    {
        DomainName = domain
    });
    return result.Availability.Value;
}
```

- For API details, see [CheckDomainAvailability](#) in *AWS SDK for .NET API Reference*.

### Check domain transferability

The following code example shows how to check the transferability of a domain.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Check the transferability of a domain name.
/// </summary>
/// <param name="domain">The domain to check for transferability.</param>
/// <returns>A transferability result string.</returns>
public async Task<string> CheckDomainTransferability(string domain)
{
    var result = await _amazonRoute53Domains.CheckDomainTransferabilityAsync(
        new CheckDomainTransferabilityRequest
    {
        DomainName = domain
    });
}
```

```
        );
    return result.Transferability.Transferable.Value;
}
```

- For API details, see [CheckDomainTransferability](#) in *AWS SDK for .NET API Reference*.

## Get domain details

The following code example shows how to get the details for a domain.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Get details for a domain.
///</summary>
///<returns>A string with detail information about the domain.</returns>
public async Task<string> GetDomainDetail(string domainName)
{
    try
    {
        var result = await _amazonRoute53Domains.GetDomainDetailAsync(
            new GetDomainDetailRequest()
            {
                DomainName = domainName
            });
        var details = $"\\tDomain {domainName}:\\n" +
                     $"\\tCreated on {result.CreationDate.ToShortDateString()}\\.\\n"
+
                     $"\\tAdmin contact is {result.AdminContact.Email}.\\n" +
                     $"\\tAuto-renew is {result.AutoRenew}.\\n";

        return details;
    }
    catch (InvalidInputException)
    {
        return $"Domain {domainName} was not found in your account.";
    }
}
```

- For API details, see [GetDomainDetail](#) in *AWS SDK for .NET API Reference*.

## Get operation details

The following code example shows how to get details on an operation.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get details for a domain action operation.
/// </summary>
/// <param name="operationId">The operational Id.</param>
/// <returns>A string describing the operational details.</returns>
public async Task<string> GetOperationDetail(string? operationId)
{
    if (operationId == null)
        return "Unable to get operational details because ID is null.";
    try
    {
        var operationDetails =
            await _amazonRoute53Domains.GetOperationDetailAsync(
                new GetOperationDetailRequest
                {
                    OperationId = operationId
                }
            );

        var details = $"\\tOperation {operationId}:\n" +
                     $"\\tFor domain {operationDetails.DomainName} on\n" +
                     $"{operationDetails.SubmittedDate.ToShortDateString()}.\n" +
                     $"\\tMessage is {operationDetails.Message}.\n" +
                     $"\\tStatus is {operationDetails.Status}.\n";

        return details;
    }
    catch (AmazonRoute53DomainsException ex)
    {
        return $"Unable to get operation details. Here's why: {ex.Message}.";
    }
}
```

- For API details, see [GetOperationDetail](#) in *AWS SDK for .NET API Reference*.

## Get suggested domain names

The following code example shows how to get domain name suggestions.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get a list of suggestions for a given domain.
/// </summary>
/// <param name="domain">The domain to check for suggestions.</param>
/// <param name="onlyAvailable">If true, only returns available domains.</param>
/// <param name="suggestionCount">The number of suggestions to return. Defaults to
the max of 50.</param>
/// <returns>A collection of domain suggestions.</returns>
public async Task<List<DomainSuggestion>> GetDomainSuggestions(string domain, bool
onlyAvailable, int suggestionCount = 50)
{
    var result = await _amazonRoute53Domains.GetDomainSuggestionsAsync(
        new GetDomainSuggestionsRequest
        {
            DomainName = domain,
            OnlyAvailable = onlyAvailable,
```

```
        SuggestionCount = suggestionCount
    }
}
return result.SuggestionsList;
}
```

- For API details, see [GetDomainSuggestions](#) in *AWS SDK for .NET API Reference*.

## List domain prices

The following code example shows how to list domain prices.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// List prices for domain type operations.
/// </summary>
/// <param name="domainTypes">Domain types to include in the results.</param>
/// <returns>The list of domain prices.</returns>
public async Task<List<DomainPrice>> ListPrices(List<string> domainTypes)
{
    var results = new List<DomainPrice>();
    var paginatePrices = _amazonRoute53Domains.Paginator.ListPrices(new
ListPricesRequest());
    // Get the entire list using the paginator.
    await foreach (var prices in paginatePrices.Prices)
    {
        results.Add(prices);
    }
    return results.Where(p => domainTypes.Contains(p.Name)).ToList();
}
```

- For API details, see [ListPrices](#) in *AWS SDK for .NET API Reference*.

## List domains

The following code example shows how to list the registered domains.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// List the domains for the account.
/// </summary>
/// <returns>A collection of domain summary records.</returns>
public async Task<List<DomainSummary>> ListDomains()
{
    var results = new List<DomainSummary>();
```

```
var paginateDomains = _amazonRoute53Domains.Paginator.ListDomains(  
    new ListDomainsRequest());  
  
// Get the entire list using the paginator.  
await foreach (var domain in paginateDomains.Domains)  
{  
    results.Add(domain);  
}  
return results;  
}
```

- For API details, see [ListDomains](#) in *AWS SDK for .NET API Reference*.

## List operations

The following code example shows how to list operations.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// List operations for the account that are submitted after a specified date.  
/// </summary>  
/// <returns>A collection of operation summary records.</returns>  
public async Task<List<OperationSummary>> ListOperations(DateTime submittedSince)  
{  
    var results = new List<OperationSummary>();  
    var paginateOperations = _amazonRoute53Domains.Paginator.ListOperations(  
        new ListOperationsRequest()  
    {  
        SubmittedSince = submittedSince  
    });  
  
    // Get the entire list using the paginator.  
    await foreach (var operations in paginateOperations.Operations)  
    {  
        results.Add(operations);  
    }  
    return results;  
}
```

- For API details, see [ListOperations](#) in *AWS SDK for .NET API Reference*.

## Register a domain

The following code example shows how to register a domain.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    ///<summary>
    ///<summary> Initiate a domain registration request.
    ///</summary>
    ///<param name="contact">Contact details.</param>
    ///<param name="domainName">The domain name to register.</param>
    ///<param name="autoRenew">True if the domain should automatically renew.</param>
    ///<param name="duration">The duration in years for the domain registration.</param>
    ///<returns>The operation Id.</returns>
    public async Task<string?> RegisterDomain(string domainName, bool autoRenew, int duration, ContactDetail contact)
    {
        // This example uses the same contact information for admin, registrant, and tech contacts.
        try
        {
            var result = await _amazonRoute53Domains.RegisterDomainAsync(
                new RegisterDomainRequest()
                {
                    AdminContact = contact,
                    RegistrantContact = contact,
                    TechContact = contact,
                    DomainName = domainName,
                    AutoRenew = autoRenew,
                    DurationInYears = duration,
                    PrivacyProtectAdminContact = false,
                    PrivacyProtectRegistrantContact = false,
                    PrivacyProtectTechContact = false
                }
            );
            return result.OperationId;
        }
        catch (InvalidOperationException)
        {
            _logger.LogInformation($"Unable to request registration for domain {domainName}");
            return null;
        }
    }
}
```

- For API details, see [RegisterDomain in AWS SDK for .NET API Reference](#).

## [View billing](#)

The following code example shows how to view billing records.

### **AWS SDK for .NET**

#### **Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    ///<summary>
    ///<summary> View billing records for the account between a start and end date.
    ///</summary>
    ///<param name="startDate">The start date for billing results.</param>
    ///<param name="endDate">The end date for billing results.</param>
    ///<returns>A collection of billing records.</returns>
    public async Task<List<BillingRecord>> ViewBilling(DateTime startDate, DateTime endDate)
    {
```

```
var results = new List<BillingRecord>();
var paginateBilling = _amazonRoute53Domains.Paginator.ViewBilling(
    new ViewBillingRequest()
{
    Start = startDate,
    End = endDate
});

// Get the entire list using the paginator.
await foreach (var billingRecords in paginateBilling.BillingRecords)
{
    results.Add(billingRecords);
}
return results;
}
```

- For API details, see [ViewBilling](#) in *AWS SDK for .NET API Reference*.

## Scenarios

### Get started with domains

The following code example shows how to:

- List current domains.
- List operations in the past year.
- View billing for the account in the past year.
- View prices for domain types.
- Get domain suggestions.
- Check domain availability.
- Check domain transferability.
- Optionally, request a domain registration.
- Get an operation detail.
- Optionally, get a domain detail.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
public static class Route53DomainScenario
{
    /*
        Before running this .NET code example, set up your development environment,
        including your credentials.

        This .NET example performs the following tasks:
        1. List current domains.
        2. List operations in the past year.
        3. View billing for the account in the past year.
        4. View prices for domain types.
        5. Get domain suggestions.
        6. Check domain availability.
    */
}
```

```
    7. Check domain transferability.
    8. Optionally, request a domain registration.
    9. Get an operation detail.
   10. Optionally, get a domain detail.
*/
private static Route53Wrapper _route53Wrapper = null!;
private static IConfiguration _configuration = null!;

static async Task Main(string[] args)
{
    // Set up dependency injection for the Amazon service.
    using var host = Host.CreateDefaultBuilder(args)
        .ConfigureLogging(logging =>
            logging.AddFilter("System", LogLevel.Debug)
                .AddFilter<DebugLoggerProvider>("Microsoft", LogLevel.Information)
                .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))
        .ConfigureServices(_,& services) =>
            services.AddAWSService<IAmazonRoute53Domains>()
                .AddTransient<Route53Wrapper>()
        )
    .Build();

    _configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load settings from .json file.
        .AddJsonFile("settings.local.json",
            true) // Optionally, load local settings.
        .Build();

    var logger = LoggerFactory.Create(builder =>
    {
        builder.AddConsole();
    }).CreateLogger(typeof(Route53DomainScenario));

    _route53Wrapper = host.Services.GetRequiredService<Route53Wrapper>();

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Welcome to the Amazon Route 53 domains example scenario.");
    Console.WriteLine(new string('-', 80));

    try
    {
        await ListDomains();
        await ListOperations();
        await ListBillingRecords();
        await ListPrices();
        await ListDomainSuggestions();
        await CheckDomainAvailability();
        await CheckDomainTransferability();
        var operationId = await RequestDomainRegistration();
        await GetOperationalDetail(operationId);
        await GetDomainDetails();
    }
    catch (Exception ex)
    {
        logger.LogError(ex, "There was a problem executing the scenario.");
    }

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("The Amazon Route 53 domains example scenario is complete.");
    Console.WriteLine(new string('-', 80));
}

/// <summary>
/// List account registered domains.

```

```
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDomains()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. List account domains.");
        var domains = await _route53Wrapper.ListDomains();
        for (int i = 0; i < domains.Count; i++)
        {
            Console.WriteLine($"{i + 1}. {domains[i].DomainName}");
        }

        if (!domains.Any())
        {
            Console.WriteLine("\tNo domains found in this account.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List domain operations in the past year.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListOperations()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. List account domain operations in the past year.");
        var operations = await _route53Wrapper.ListOperations(
            DateTime.Today.AddYears(-1));
        for (int i = 0; i < operations.Count; i++)
        {
            Console.WriteLine($" \tOperation Id: {operations[i].OperationId}");
            Console.WriteLine($" \tStatus: {operations[i].Status}");
            Console.WriteLine($" \tDate: {operations[i].SubmittedDate}");
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List billing in the past year.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListBillingRecords()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. View billing for the account in the past year.");
        var billingRecords = await _route53Wrapper.ViewBilling(
            DateTime.Today.AddYears(-1),
            DateTime.Today);
        for (int i = 0; i < billingRecords.Count; i++)
        {
            Console.WriteLine($" \tBill Date:");
            Console.WriteLine($"{billingRecords[i].BillDate.ToShortDateString()}");
            Console.WriteLine($" \tOperation: {billingRecords[i].Operation}");
            Console.WriteLine($" \tPrice: {billingRecords[i].Price}");
        }
        if (!billingRecords.Any())
        {
            Console.WriteLine("\tNo billing records found in this account for the past
year.");
        }
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
```

```
    /// List prices for a few domain types.  
    /// </summary>  
    /// <returns>Async task.</returns>  
    private static async Task ListPrices()  
{  
    Console.WriteLine(new string('-', 80));  
    Console.WriteLine($"4. View prices for domain types.");  
    var domainTypes = new List<string> { "net", "com", "org", "co" };  
  
    var prices = await _route53Wrapper.ListPrices(domainTypes);  
    foreach (var pr in prices)  
    {  
        Console.WriteLine($"\\tName: {pr.Name}");  
        Console.WriteLine($"\\tRegistration: {pr.RegistrationPrice?.Price}  
{pr.RegistrationPrice?.Currency}");  
        Console.WriteLine($"\\tRenewal: {pr.RenewalPrice?.Price}  
{pr.RenewalPrice?.Currency}");  
        Console.WriteLine($"\\tTransfer: {pr.TransferPrice?.Price}  
{pr.TransferPrice?.Currency}");  
        Console.WriteLine($"\\tChange Ownership: {pr.ChangeOwnershipPrice?.Price}  
{pr.ChangeOwnershipPrice?.Currency}");  
        Console.WriteLine($"\\tRestoration: {pr.RestorationPrice?.Price}  
{pr.RestorationPrice?.Currency}");  
        Console.WriteLine();  
    }  
    Console.WriteLine(new string('-', 80));  
}  
  
    /// <summary>  
    /// List domain suggestions for a domain name.  
    /// </summary>  
    /// <returns>Async task.</returns>  
    private static async Task ListDomainSuggestions()  
{  
    Console.WriteLine(new string('-', 80));  
    Console.WriteLine($"5. Get domain suggestions.");  
    string? domainName = null;  
    while (domainName == null || string.IsNullOrWhiteSpace(domainName))  
    {  
        Console.WriteLine($"Enter a domain name to get available domain  
suggestions.");  
        domainName = Console.ReadLine();  
    }  
  
    var suggestions = await _route53Wrapper.GetDomainSuggestions(domainName, true,  
5);  
    foreach (var suggestion in suggestions)  
    {  
        Console.WriteLine($"\\tSuggestion Name: {suggestion.DomainName}");  
        Console.WriteLine($"\\tAvailability: {suggestion.Availability}");  
    }  
    Console.WriteLine(new string('-', 80));  
}  
  
    /// <summary>  
    /// Check availability for a domain name.  
    /// </summary>  
    /// <returns>Async task.</returns>  
    private static async Task CheckDomainAvailability()  
{  
    Console.WriteLine(new string('-', 80));  
    Console.WriteLine($"6. Check domain availability.");  
    string? domainName = null;  
    while (domainName == null || string.IsNullOrWhiteSpace(domainName))  
    {  
        Console.WriteLine($"Enter a domain name to check domain availability.");  
    }  
}
```

```
        domainName = Console.ReadLine();
    }

    var availability = await _route53Wrapper.CheckDomainAvailability(domainName);
    Console.WriteLine($"\\tAvailability: {availability}");
    Console.WriteLine(new string('-', 80));
}

/// <summary>
/// Check transferability for a domain name.
/// </summary>
/// <returns>Async task.</returns>
private static async Task CheckDomainTransferability()
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"7. Check domain transferability.");
    string? domainName = null;
    while (domainName == null || string.IsNullOrWhiteSpace(domainName))
    {
        Console.WriteLine($"Enter a domain name to check domain transferability.");
        domainName = Console.ReadLine();
    }

    var transferability = await
_route53Wrapper.CheckDomainTransferability(domainName);
    Console.WriteLine($"\\tTransferability: {transferability}");

    Console.WriteLine(new string('-', 80));
}

/// <summary>
/// Check transferability for a domain name.
/// </summary>
/// <returns>Async task.</returns>
private static async Task<string?> RequestDomainRegistration()
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"8. Optionally, request a domain registration.");

    Console.WriteLine($"\\tNote: This example uses domain request settings in
settings.json.");
    Console.WriteLine($"\\tTo change the domain registration settings, set the
values in that file.");
    Console.WriteLine($"\\tRemember, registering an actual domain will incur an
account billing cost.");
    Console.WriteLine($"\\tWould you like to begin a domain registration? (y/n)");
    var ynResponse = Console.ReadLine();
    if (ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase))
    {
        string domainName = _configuration["DomainName"];
        ContactDetail contact = new ContactDetail();
        contact.CountryCode =
CountryCode.FindValue(_configuration["Contact:CountryCode"]);
        contact.ContactType =
ContactType.FindValue(_configuration["Contact:ContactType"]);

        _configuration.GetSection("Contact").Bind(contact);

        var operationId = await _route53Wrapper.RegisterDomain(
            domainName,
            Convert.ToBoolean(_configuration["AutoRenew"]),
            Convert.ToInt32(_configuration["DurationInYears"]),
            contact);
        if (operationId != null)
        {
    
```

```

        Console.WriteLine(
            $"\\tRegistration requested. Operation Id: {operationId}");
    }

    return operationId;
}

Console.WriteLine(new string('-', 80));
return null;
}

/// <summary>
/// Get details for an operation.
/// </summary>
/// <returns>Async task.</returns>
private static async Task GetOperationalDetail(string? operationId)
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"9. Get an operation detail.");

    var operationDetails =
        await _route53Wrapper.GetOperationDetail(operationId);

    Console.WriteLine(operationDetails);

    Console.WriteLine(new string('-', 80));
}

/// <summary>
/// Optionally, get details for a registered domain.
/// </summary>
/// <returns>Async task.</returns>
private static async Task<string?> GetDomainDetails()
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"10. Get details on a domain.");

    Console.WriteLine($"\\tNote: you must have a registered domain to get
details.");
    Console.WriteLine($"\\tWould you like to get domain details? (y/n)");
    var ynResponse = Console.ReadLine();
    if (ynResponse != null && ynResponse.Equals("y",
 StringComparison.InvariantCultureIgnoreCase))
    {
        string? domainName = null;
        while (domainName == null)
        {
            Console.WriteLine($"\\tEnter a domain name to get details.");
            domainName = Console.ReadLine();
        }

        var domainDetails = await _route53Wrapper.GetDomainDetail(domainName);
        Console.WriteLine(domainDetails);
    }

    Console.WriteLine(new string('-', 80));
    return null;
}
}

```

Wrapper methods used by the scenario for Route 53 domain registration actions.

```
public class Route53Wrapper
```

```
{  
    private readonly IAmazonRoute53Domains _amazonRoute53Domains;  
    private readonly ILogger<Route53Wrapper> _logger;  
    public Route53Wrapper(IAmazonRoute53Domains amazonRoute53Domains,  
    ILogger<Route53Wrapper> logger)  
    {  
        _amazonRoute53Domains = amazonRoute53Domains;  
        _logger = logger;  
    }  
  
    ///<summary>  
    /// List prices for domain type operations.  
    ///</summary>  
    ///<param name="domainTypes">Domain types to include in the results.</param>  
    ///<returns>The list of domain prices.</returns>  
    public async Task<List<DomainPrice>> ListPrices(List<string> domainTypes)  
    {  
        var results = new List<DomainPrice>();  
        var paginatePrices = _amazonRoute53Domains.Paginator.ListPrices(new  
ListPricesRequest());  
        // Get the entire list using the paginator.  
        await foreach (var prices in paginatePrices.Prices)  
        {  
            results.Add(prices);  
        }  
        return results.Where(p => domainTypes.Contains(p.Name)).ToList();  
    }  
  
    ///<summary>  
    /// Check the availability of a domain name.  
    ///</summary>  
    ///<param name="domain">The domain to check for availability.</param>  
    ///<returns>An availability result string.</returns>  
    public async Task<string> CheckDomainAvailability(string domain)  
    {  
        var result = await _amazonRoute53Domains.CheckDomainAvailabilityAsync(  
            new CheckDomainAvailabilityRequest  
            {  
                DomainName = domain  
            }  
        );  
        return result.Availability.Value;  
    }  
  
    ///<summary>  
    /// Check the transferability of a domain name.  
    ///</summary>  
    ///<param name="domain">The domain to check for transferability.</param>  
    ///<returns>A transferability result string.</returns>  
    public async Task<string> CheckDomainTransferability(string domain)  
    {  
        var result = await _amazonRoute53Domains.CheckDomainTransferabilityAsync(  
            new CheckDomainTransferabilityRequest  
            {  
                DomainName = domain  
            }  
        );  
        return result.Transferability.Transferable.Value;  
    }  
  
    ///<summary>  
    /// Get a list of suggestions for a given domain.  
    ///</summary>
```

```

    ///</summary>
    ///<param name="domain">The domain to check for suggestions.</param>
    ///<param name="onlyAvailable">If true, only returns available domains.</param>
    ///<param name="suggestionCount">The number of suggestions to return. Defaults to
    the max of 50.</param>
    ///<returns>A collection of domain suggestions.</returns>
    public async Task<List<DomainSuggestion>> GetDomainSuggestions(string domain, bool
onlyAvailable, int suggestionCount = 50)
    {
        var result = await _amazonRoute53Domains.GetDomainSuggestionsAsync(
            new GetDomainSuggestionsRequest
            {
                DomainName = domain,
                OnlyAvailable = onlyAvailable,
                SuggestionCount = suggestionCount
            }
        );
        return result.SuggestionsList;
    }

    ///<summary>
    /// Get details for a domain action operation.
    ///</summary>
    ///<param name="operationId">The operational Id.</param>
    ///<returns>A string describing the operational details.</returns>
    public async Task<string> GetOperationDetail(string? operationId)
    {
        if (operationId == null)
            return "Unable to get operational details because ID is null.";
        try
        {
            var operationDetails =
                await _amazonRoute53Domains.GetOperationDetailAsync(
                    new GetOperationDetailRequest
                    {
                        OperationId = operationId
                    }
                );

            var details = $"\\tOperation {operationId}:\n" +
                $"\\tFor domain {operationDetails.DomainName} on\n{operationDetails.SubmittedDate.ToShortDateString()}.\\n" +
                $"\\tMessage is {operationDetails.Message}.\\n" +
                $"\\tStatus is {operationDetails.Status}.\\n";

            return details;
        }
        catch (AmazonRoute53DomainsException ex)
        {
            return $"Unable to get operation details. Here's why: {ex.Message}.";
        }
    }

    ///<summary>
    /// Initiate a domain registration request.
    ///</summary>
    ///<param name="contact">Contact details.</param>
    ///<param name="domainName">The domain name to register.</param>
    ///<param name="autoRenew">True if the domain should automatically renew.</param>
    ///<param name="duration">The duration in years for the domain registration.</
param>
    ///<returns>The operation Id.</returns>
    public async Task<string?> RegisterDomain(string domainName, bool autoRenew, int
duration, ContactDetail contact)

```

```
{  
    // This example uses the same contact information for admin, registrant, and  
    tech contacts.  
    try  
    {  
        var result = await _amazonRoute53Domains.RegisterDomainAsync(  
            new RegisterDomainRequest()  
        {  
            AdminContact = contact,  
            RegistrantContact = contact,  
            TechContact = contact,  
            DomainName = domainName,  
            AutoRenew = autoRenew,  
            DurationInYears = duration,  
            PrivacyProtectAdminContact = false,  
            PrivacyProtectRegistrantContact = false,  
            PrivacyProtectTechContact = false  
        }  
    );  
    return result.OperationId;  
}  
catch (InvalidOperationException)  
{  
    _logger.LogInformation($"Unable to request registration for domain  
{domainName}");  
    return null;  
}  
}  
  
/// <summary>  
/// View billing records for the account between a start and end date.  
/// </summary>  
/// <param name="startDate">The start date for billing results.</param>  
/// <param name="endDate">The end date for billing results.</param>  
/// <returns>A collection of billing records.</returns>  
public async Task<List<BillingRecord>> ViewBilling(DateTime startDate, DateTime  
endDate)  
{  
    var results = new List<BillingRecord>();  
    var paginateBilling = _amazonRoute53Domains.Paginator.ViewBilling(  
        new ViewBillingRequest()  
    {  
        Start = startDate,  
        End = endDate  
    });  
  
    // Get the entire list using the paginator.  
    await foreach (var billingRecords in paginateBilling.BillingRecords)  
    {  
        results.Add(billingRecords);  
    }  
    return results;  
}  
  
/// <summary>  
/// List the domains for the account.  
/// </summary>  
/// <returns>A collection of domain summary records.</returns>  
public async Task<List<DomainSummary>> ListDomains()  
{  
    var results = new List<DomainSummary>();  
    var paginateDomains = _amazonRoute53Domains.Paginator.ListDomains(  
        new ListDomainsRequest());
```

```

        // Get the entire list using the paginator.
        await foreach (var domain in paginateDomains.Domains)
        {
            results.Add(domain);
        }
        return results;
    }

    /// <summary>
    /// List operations for the account that are submitted after a specified date.
    /// </summary>
    /// <returns>A collection of operation summary records.</returns>
    public async Task<List<OperationSummary>> ListOperations(DateTime submittedSince)
    {
        var results = new List<OperationSummary>();
        var paginateOperations = _amazonRoute53Domains.Paginator.ListOperations(
            new ListOperationsRequest()
            {
                SubmittedSince = submittedSince
            });

        // Get the entire list using the paginator.
        await foreach (var operations in paginateOperations.Operations)
        {
            results.Add(operations);
        }
        return results;
    }

    /// <summary>
    /// Get details for a domain.
    /// </summary>
    /// <returns>A string with detail information about the domain.</returns>
    public async Task<string> GetDomainDetail(string domainName)
    {
        try
        {
            var result = await _amazonRoute53Domains.GetDomainDetailAsync(
                new GetDomainDetailRequest()
                {
                    DomainName = domainName
                });
            var details = $"{\tDomain {domainName}}:\n" +
                $"{\tCreated on {result.CreationDate.ToShortDateString()}}.\n"
+
                $"{\tAdmin contact is {result.AdminContact.Email}}.\n" +
                $"{\tAuto-renew is {result.AutoRenew}}.\n";

            return details;
        }
        catch (InvalidArgumentException)
        {
            return $"Domain {domainName} was not found in your account.";
        }
    }
}

```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.

- [CheckDomainAvailability](#)
- [CheckDomainTransferability](#)
- [GetDomainDetail](#)

- [GetDomainSuggestions](#)
- [GetOperationDetail](#)
- [ListDomains](#)
- [ListOperations](#)
- [ListPrices](#)
- [RegisterDomain](#)
- [ViewBilling](#)

## Amazon S3 examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon Simple Storage Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2516\)](#)
- [Scenarios \(p. 2522\)](#)

## Actions

### Copy an object from one bucket to another

The following code example shows how to copy an S3 object from one bucket to another.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Copies an object in an Amazon S3 bucket to a folder within the
/// same bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
/// object to copy is located.</param>
/// <param name="objectName">The object to be copied.</param>
/// <param name="folderName">The folder to which the object will
/// be copied.</param>
/// <returns>A boolean value that indicates the success or failure of
/// the copy operation.</returns>
public static async Task<bool> CopyObjectInBucketAsync(
    IAmazonS3 client,
    string bucketName,
    string objectName,
    string folderName)
{
    try
    {
        var request = new CopyObjectRequest
```

```
        {
            SourceBucket = bucketName,
            SourceKey = objectName,
            DestinationBucket = bucketName,
            DestinationKey = $"{folderName}\\{objectName}",
        };
        var response = await client.CopyObjectAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error copying object: '{ex.Message}'");
        return false;
    }
}
```

- For API details, see [CopyObject](#) in *AWS SDK for .NET API Reference*.

## Create a bucket

The following code example shows how to create an S3 bucket.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Shows how to create a new Amazon S3 bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>A boolean value representing the success or failure of
/// the bucket creation process.</returns>
public static async Task<bool> CreateBucketAsync(IAmazonS3 client, string
bucketName)
{
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
        };

        var response = await client.PutBucketAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for .NET API Reference*.

## Delete an empty bucket

The following code example shows how to delete an empty S3 bucket.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Shows how to delete an Amazon S3 bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket to delete.</param>
/// <returns>A boolean value that represents the success or failure of
/// the delete operation.</returns>
public static async Task<bool> DeleteBucketAsync(IAmazonS3 client, string
bucketName)
{
    var request = new DeleteBucketRequest
    {
        BucketName = bucketName,
    };

    var response = await client.DeleteBucketAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- For API details, see [DeleteBucket](#) in *AWS SDK for .NET API Reference*.

## Delete multiple objects

The following code example shows how to delete multiple objects from an S3 bucket.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete all objects in an S3 bucket.

```
/// <summary>
/// Delete all of the objects stored in an existing Amazon S3 bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The name of the bucket from which the
/// contents will be deleted.</param>
/// <returns>A boolean value that represents the success or failure of
/// deleting all of the objects in the bucket.</returns>
public static async Task<bool> DeleteBucketContentsAsync(IAmazonS3 client,
string bucketName)
{
    // Iterate over the contents of the bucket and delete all objects.
```

```
var request = new ListObjectsV2Request
{
    BucketName = bucketName,
};

try
{
    var response = await client.ListObjectsV2Async(request);

    do
    {
        response.S3Objects
            .ForEach(async obj => await
client.DeleteObjectAsync(bucketName, obj.Key));

        // If the response is truncated, set the request ContinuationToken
        // from the NextContinuationToken property of the response.
        request.ContinuationToken = response.NextContinuationToken;
    }
    while (response.IsTruncated);

    return true;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error deleting objects: {ex.Message}");
    return false;
}
}
```

- For API details, see [DeleteObjects](#) in *AWS SDK for .NET API Reference*.

## Get an object from a bucket

The following code example shows how to read data from an object in an S3 bucket.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Shows how to download an object from an Amazon S3 bucket to the
/// local computer.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The name of the bucket where the object is
/// currently stored.</param>
/// <param name="objectName">The name of the object to download.</param>
/// <param name="filePath">The path, including filename, where the
/// downloaded object will be stored.</param>
/// <returns>A boolean value indicating the success or failure of the
/// download process.</returns>
public static async Task<bool> DownloadObjectFromBucketAsync(
    IAmazonS3 client,
    string bucketName,
    string objectName,
    string filePath)
```

```
{  
    // Create a GetObject request  
    var request = new GetObjectRequest  
    {  
        BucketName = bucketName,  
        Key = objectName,  
    };  
  
    // Issue request and remember to dispose of the response  
    using GetObjectResponse response = await client.GetObjectAsync(request);  
  
    try  
    {  
        // Save object to local file  
        await response.WriteResponseStreamToFileAsync($"{filePath}\  
\{objectName}", true, System.Threading.CancellationToken.None);  
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;  
    }  
    catch (AmazonS3Exception ex)  
    {  
        Console.WriteLine($"Error saving {objectName}: {ex.Message}");  
        return false;  
    }  
}
```

- For API details, see [GetObject](#) in *AWS SDK for .NET API Reference*.

## List objects in a bucket

The following code example shows how to list objects in an S3 bucket.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Shows how to list the objects in an Amazon S3 bucket.  
/// </summary>  
/// <param name="client">An initialized Amazon S3 client object.</param>  
/// <param name="bucketName">The name of the bucket for which to list  
/// the contents.</param>  
/// <returns>A boolean value indicating the success or failure of the  
/// copy operation.</returns>  
public static async Task<bool> ListBucketContentsAsync(IAmazonS3 client, string  
bucketName)  
{  
    try  
    {  
        var request = new ListObjectsV2Request  
        {  
            BucketName = bucketName,  
            MaxKeys = 5,  
        };  
  
        Console.WriteLine("-----");  
        Console.WriteLine($"Listing the contents of {bucketName}:");  
        Console.WriteLine("-----");  
    }
```

```
        var response = new ListObjectsV2Response();

        do
        {
            response = await client.ListObjectsV2Async(request);

            response.S3Objects
                .ForEach(obj => Console.WriteLine($"{obj.Key,-35}{obj.LastModified.ToShortDateString(),10}{obj.Size,10}"));

            // If the response is truncated, set the request ContinuationToken
            // from the NextContinuationToken property of the response.
            request.ContinuationToken = response.NextContinuationToken;
        }
        while (response.IsTruncated);

        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error encountered on server. Message:{ex.Message}'");
        getting list of objects.");
        return false;
    }
}
```

- For API details, see [ListObjects](#) in *AWS SDK for .NET API Reference*.

## Upload an object to a bucket

The following code example shows how to upload an object to an S3 bucket.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Shows how to upload a file from the local computer to an Amazon S3
/// bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client object.</param>
/// <param name="bucketName">The Amazon S3 bucket to which the object
/// will be uploaded.</param>
/// <param name="objectName">The object to upload.</param>
/// <param name="filePath">The path, including file name, of the object
/// on the local computer to upload.</param>
/// <returns>A boolean value indicating the success or failure of the
/// upload procedure.</returns>
public static async Task<bool> UploadFileAsync(
    IAmazonS3 client,
    string bucketName,
    string objectName,
    string filePath)
{
    var request = new PutObjectRequest
    {
```

```
        BucketName = bucketName,
        Key = objectName,
        FilePath = filePath,
    };

    var response = await client.PutObjectAsync(request);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"Successfully uploaded {objectName} to {bucketName}.");
        return true;
    }
    else
    {
        Console.WriteLine($"Could not upload {objectName} to {bucketName}.");
        return false;
    }
}
```

- For API details, see [PutObject](#) in *AWS SDK for .NET API Reference*.

## Scenarios

### Get started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.
- Delete a bucket.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;

public class S3_Basics
{
    public static async Task Main()
    {
        // Create an Amazon S3 client object. The constructor uses the
        // default user installed on the system. To work with Amazon S3
        // features in a different AWS Region, pass the AWS Region as a
        // parameter to the client constructor.
        IAmazonS3 client = new AmazonS3Client();
```

```
string bucketName = string.Empty;
string filePath = string.Empty;
string keyName = string.Empty;

Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
Console.WriteLine("procedures. This application will:");
Console.WriteLine("\n\t1. Create a bucket");
Console.WriteLine("\n\t2. Upload an object to the new bucket");
Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
bucket");
Console.WriteLine("\n\t4. List the items in the new bucket");
Console.WriteLine("\n\t5. Delete all the items in the bucket");
Console.WriteLine("\n\t6. Delete the bucket");

Console.WriteLine("-----");

// Create a bucket.
Console.WriteLine("\nCreate a new Amazon S3 bucket.\n");

Console.Write("Please enter a name for the new bucket: ");
bucketName = Console.ReadLine();

var success = await S3Bucket.CreateBucketAsync(client, bucketName);
if (success)
{
    Console.WriteLine($"Successfully created bucket: {bucketName}.\n");
}
else
{
    Console.WriteLine($"Could not create bucket: {bucketName}.\n");
}

Console.WriteLine("Upload a file to the new bucket.");

// Get the local path and filename for the file to upload.
while (string.IsNullOrEmpty(filePath))
{
    Console.Write("Please enter the path and filename of the file to
upload: ");
    filePath = Console.ReadLine();

    // Confirm that the file exists on the local computer.
    if (!File.Exists(filePath))
    {
        Console.WriteLine($"Couldn't find {filePath}. Try again.\n");
        filePath = string.Empty;
    }
}

// Get the file name from the full path.
keyName = Path.GetFileName(filePath);

success = await S3Bucket.UploadFileAsync(client, bucketName, keyName,
filePath);

if (success)
{
    Console.WriteLine($"Successfully uploaded {keyName} from {filePath} to
{bucketName}.\n");
}
else
{
    Console.WriteLine($"Could not upload {keyName}.\n");
}

// Set the file path to an empty string to avoid overwriting the
```

```
// file we just uploaded to the bucket.  
filePath = string.Empty;  
  
// Now get a new location where we can save the file.  
while (string.IsNullOrEmpty(filePath))  
{  
    // First get the path to which the file will be downloaded.  
    Console.WriteLine("Please enter the path where the file will be downloaded:  
");  
    filePath = Console.ReadLine();  
  
    // Confirm that the file exists on the local computer.  
    if (File.Exists($"{filePath}\\{keyName}"))  
    {  
        Console.WriteLine($"Sorry, the file already exists in that  
location.\n");  
        filePath = string.Empty;  
    }  
}  
  
// Download an object from a bucket.  
success = await S3Bucket.DownloadObjectFromBucketAsync(client, bucketName,  
keyName, filePath);  
  
if (success)  
{  
    Console.WriteLine($"Successfully downloaded {keyName}.\n");  
}  
else  
{  
    Console.WriteLine($"Sorry, could not download {keyName}.\n");  
}  
  
// Copy the object to a different folder in the bucket.  
string folderName = string.Empty;  
  
while (string.IsNullOrEmpty(folderName))  
{  
    Console.WriteLine("Please enter the name of the folder to copy your object  
to: ");  
    folderName = Console.ReadLine();  
}  
  
while (string.IsNullOrEmpty(keyName))  
{  
    // Get the name to give to the object once uploaded.  
    Console.WriteLine("Enter the name of the object to copy: ");  
    keyName = Console.ReadLine();  
}  
  
await S3Bucket.CopyObjectInBucketAsync(client, bucketName, keyName,  
folderName);  
  
// List the objects in the bucket.  
await S3Bucket.ListBucketContentsAsync(client, bucketName);  
  
// Delete the contents of the bucket.  
await S3Bucket.DeleteBucketContentsAsync(client, bucketName);  
  
// Deleting the bucket too quickly after deleting its contents will  
// cause an error that the bucket isn't empty. So...  
Console.WriteLine("Press <Enter> when you are ready to delete the  
bucket.");  
_ = Console.ReadLine();  
  
// Delete the bucket.
```

```
        await S3Bucket.DeleteBucketAsync(client, bucketName);
    }
}
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

## Upload or download large files

The following code example shows how to upload or download large files to and from Amazon S3.

For more information, see [Uploading an object using multipart upload](#).

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Call functions that transfer files to and from an S3 bucket using the Amazon S3 TransferUtility.

```
global using System.Text;
global using Amazon;
global using Amazon.S3;
global using Amazon.S3.Model;
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;

// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
 IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
    $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\TransferFolder";
```

```
DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.`);

var success = await TransferMethods.UploadSingleFileAsync(transferUtil, bucketName,
    fileToUpload, localPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
    {bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil, bucketName,
    keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
    {bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDelete"];
Console.WriteLine($"Downloading {keyName} from {bucketName}.`);

success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
    keyName, localPath);
if (success)
{
    Console.WriteLine($"Successfully downloaded the file, {keyName} from
    {bucketName}.");
}

PressEnter();
```

```
// Download the contents of a directory from an S3 bucket.
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
var downloadPath = $"{localPath}\\"{s3Path}";

Console.WriteLine($"Downloading the contents of {bucketName}\\"{s3Path}");
Console.WriteLine($"{bucketName}\\"{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();

success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil, bucketName,
    s3Path, downloadPath);
if (success)
{
    Console.WriteLine($"Downloaded the files in {bucketName} to {downloadPath}.");
    Console.WriteLine($"{downloadPath} now contains the following files:");
    DisplayLocalFiles(downloadPath);
}

Console.WriteLine("\nThe TransferUtility Basics application has completed.");
PressEnter();

// Displays the title for a section of the scenario.
static void DisplayTitle(string titleText)
{
    var sepBar = new string('-', Console.WindowWidth);

    Console.WriteLine(sepBar);
    Console.WriteLine(CenterText(titleText));
    Console.WriteLine(sepBar);
}

// Displays a description of the actions to be performed by the scenario.
static void DisplayInstructions()
{
    var sepBar = new string('-', Console.WindowWidth);

    DisplayTitle("Amazon S3 Transfer Utility Basics");
    Console.WriteLine("This program shows how to use the Amazon S3 Transfer Utility.");
    Console.WriteLine("It performs the following actions:");
    Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
    Console.WriteLine("\t2. Upload an entire directory from the local computer to an\n\tS3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
    Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
directory.");
    Console.WriteLine($"{sepBar}");
}

// Pauses the scenario.
static void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\n");
}

// Returns the string textToCenter, padded on the left with spaces
// that center the text on the console display.
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth - textToCenter.Length) / 2));
    centeredText.Append(textToCenter);
```

```

        return centeredText.ToString();
    }

    // Displays a list of file names included in the specified path.
    static void DisplayLocalFiles(string localPath)
    {
        var fileList = Directory.GetFiles(localPath);
        if (fileList.Length > 0)
        {
            foreach (var fileName in fileList)
            {
                Console.WriteLine(fileName);
            }
        }
    }

    // Displays a list of the files in the specified S3 bucket and prefix.
    static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string s3Path)
    {
        ListObjectsV2Request request = new()
        {
            BucketName = bucketName,
            Prefix = s3Path,
            MaxKeys = 5,
        };

        var response = new ListObjectsV2Response();

        do
        {
            response = await client.ListObjectsV2Async(request);

            response.S3Objects
                .ForEach(obj => Console.WriteLine($"{obj.Key}"));

            // If the response is truncated, set the request ContinuationToken
            // from the NextContinuationToken property of the response.
            request.ContinuationToken = response.NextContinuationToken;
        } while (response.IsTruncated);
    }
}

```

Upload a single file.

```

///<summary>
/// Uploads a single file from the local computer to an S3 bucket.
///</summary>
///<param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
///<param name="bucketName">The name of the S3 bucket where the file
/// will be stored.</param>
///<param name="fileName">The name of the file to upload.</param>
///<param name="localPath">The local path where the file is stored.</param>
///<returns>A boolean value indicating the success of the action.</returns>
public static async Task<bool> UploadSingleFileAsync(
    TransferUtility transferUtil,
    string bucketName,
    string fileName,
    string localPath)
{
    if (File.Exists($"{localPath}\\{fileName}"))
    {

```

```
        try
        {
            await transferUtil.UploadAsync(new TransferUtilityUploadRequest
            {
                BucketName = bucketName,
                Key = fileName,
                FilePath = $"{localPath}\\{fileName}",
            });

            return true;
        }
        catch (AmazonS3Exception s3Ex)
        {
            Console.WriteLine($"Could not upload {fileName} from {localPath}
because:");
            Console.WriteLine(s3Ex.Message);
            return false;
        }
    }
    else
    {
        Console.WriteLine($"{fileName} does not exist in {localPath}");
        return false;
    }
}
```

Upload an entire local directory.

```
/// <summary>
/// Uploads all the files in a local directory to a directory in an S3
/// bucket.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
/// <param name="bucketName">The name of the S3 bucket where the files
/// will be stored.</param>
/// <param name="keyPrefix">The key prefix is the S3 directory where
/// the files will be stored.</param>
/// <param name="localPath">The local directory that contains the files
/// to be uploaded.</param>
/// <returns>A Boolean value representing the success of the action.</returns>
public static async Task<bool> UploadFullDirectoryAsync(
    TransferUtility transferUtil,
    string bucketName,
    string keyPrefix,
    string localPath)
{
    if (Directory.Exists(localPath))
    {
        try
        {
            await transferUtil.UploadDirectoryAsync(new
TransferUtilityUploadDirectoryRequest
            {
                BucketName = bucketName,
                KeyPrefix = keyPrefix,
                Directory = localPath,
            });
            return true;
        }
        catch (AmazonS3Exception s3Ex)
        {
```

```
        Console.WriteLine($"Can't upload the contents of {localPath}  
because:");
        Console.WriteLine(s3Ex?.Message);
        return false;
    }
}
else
{
    Console.WriteLine($"The directory {localPath} does not exist.");
    return false;
}
}
```

Download a single file.

```
/// <summary>
/// Download a single file from an S3 bucket to the local computer.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
/// <param name="bucketName">The name of the S3 bucket containing the
/// file to download.</param>
/// <param name="keyName">The name of the file to download.</param>
/// <param name="localPath">The path on the local computer where the
/// downloaded file will be saved.</param>
/// <returns>A Boolean value indicating the results of the action.</returns>
public static async Task<bool> DownloadSingleFileAsync(
    TransferUtility transferUtil,
    string bucketName,
    string keyName,
    string localPath)
{
    await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest
    {
        BucketName = bucketName,
        Key = keyName,
        FilePath = $"{localPath}\\{keyName}",
    });
    return (File.Exists($"{localPath}\\{keyName}"));
}
```

Download contents of an S3 bucket.

```
/// <summary>
/// Downloads the contents of a directory in an S3 bucket to a
/// directory on the local computer.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
/// <param name="bucketName">The bucket containing the files to download.</
param>
/// <param name="s3Path">The S3 directory where the files are located.</param>
/// <param name="localPath">The local path to which the files will be
/// saved.</param>
/// <returns>A Boolean value representing the success of the action.</returns>
public static async Task<bool> DownloadS3DirectoryAsync(
```

```
        TransferUtility transferUtil,
        string bucketName,
        string s3Path,
        string localPath)
    {
        int fileCount = 0;

        // If the directory doesn't exist, it will be created.
        if (Directory.Exists(s3Path))
        {
            var files = Directory.GetFiles(localPath);
            fileCount = files.Length;
        }

        await transferUtil.DownloadDirectoryAsync(new
TransferUtilityDownloadDirectoryRequest
{
    BucketName = bucketName,
    LocalDirectory = localPath,
    S3Directory = s3Path,
});

        if (Directory.Exists(localPath))
        {
            var files = Directory.GetFiles(localPath);
            if (files.Length > fileCount)
            {
                return true;
            }

            // No change in the number of files. Assume
            // the download failed.
            return false;
        }

        // The local directory doesn't exist. No files
        // were downloaded.
        return false;
    }
}
```

## S3 Glacier examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon S3 Glacier.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2531\)](#)

## Actions

### Add tags

The following code example shows how to add tags to an Amazon S3 Glacier vault.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Glacier;
using Amazon.Glacier.Model;

public class AddTagsToVault
{
    public static async Task Main(string[] args)
    {
        string vaultName = "example-vault";

        var client = new AmazonGlacierClient();
        var request = new AddTagsToVaultRequest
        {
            Tags = new Dictionary<string, string>
            {
                { "examplekey1", "examplevalue1" },
                { "examplekey2", "examplevalue2" },
            },
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await client.AddTagsToVaultAsync(request);
    }
}
```

- For API details, see [AddTagsToVault in AWS SDK for .NET API Reference](#).

## Create a vault

The following code example shows how to create an Amazon S3 Glacier vault.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Glacier;
using Amazon.Glacier.Model;

public class CreateVault
{
    static async Task Main(string[] args)
    {
        string vaultName = "example-vault";
        var client = new AmazonGlacierClient();
        var request = new CreateVaultRequest
```

```
{  
    // Setting the AccountId to "-" means that  
    // the account associated with the default  
    // client will be used.  
    AccountId = "-",  
    VaultName = vaultName,  
};  
  
var response = await client.CreateVaultAsync(request);  
  
Console.WriteLine($"Created {vaultName} at: {response.Location}");  
}  
}
```

- For API details, see [CreateVault](#) in *AWS SDK for .NET API Reference*.

## Describe a job

The following code example shows how to describe an Amazon S3 Glacier job.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Threading.Tasks;  
using Amazon.Glacier;  
using Amazon.Glacier.Model;  
  
public class DescribeVault  
{  
    public static async Task Main(string[] args)  
    {  
        string vaultName = "example-vault";  
        var client = new AmazonGlacierClient();  
        var request = new DescribeVaultRequest  
        {  
            AccountId = "-",  
            VaultName = vaultName,  
        };  
  
        var response = await client.DescribeVaultAsync(request);  
  
        // Display the information about the vault.  
        Console.WriteLine($"{response.VaultName}\tARN: {response.VaultARN}");  
        Console.WriteLine($"Created on: {response.CreationDate}\tNumber of  
Archives: {response.NumberOfArchives}\tSize (in bytes): {response.SizeInBytes}");  
        if (response.LastInventoryDate != DateTime.MinValue)  
        {  
            Console.WriteLine($"Last inventory: {response.LastInventoryDate}");  
        }  
    }  
}
```

- For API details, see [DescribeJob](#) in *AWS SDK for .NET API Reference*.

## Download an archive

The following code example shows how to download an Amazon S3 Glacier archive.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using Amazon;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;
using Amazon.Runtime;

class DownloadArchiveHighLevel
{
    private static readonly string VaultName = "examplevault";
    private static readonly string ArchiveId = "*** Provide archive ID ***";
    private static readonly string DownloadFilePath = "*** Provide the file name
and path to where to store the download ***";
    private static int currentPercentage = -1;

    static void Main()
    {
        try
        {
            var manager = new ArchiveTransferManager(RegionEndpoint.USEast2);

            var options = new DownloadOptions
            {
                StreamTransferProgress = Progress,
            };

            // Download an archive.
            Console.WriteLine("Initiating the archive retrieval job and then polling
SQS queue for the archive to be available.");
            Console.WriteLine("Once the archive is available, downloading will
begin.");
            manager.DownloadAsync(VaultName, ArchiveId, DownloadFilePath, options);
            Console.WriteLine("To continue, press Enter");
            Console.ReadKey();
        }
        catch (AmazonGlacierException ex)
        {
            Console.WriteLine(ex.Message);
        }

        Console.WriteLine("To continue, press Enter");
        Console.ReadKey();
    }

    static void Progress(object sender, StreamTransferProgressArgs args)
    {
        if (args.PercentDone != currentPercentage)
        {
            currentPercentage = args.PercentDone;
            Console.WriteLine("Downloaded {0}%", args.PercentDone);
        }
    }
}
```

## List jobs

The following code example shows how to list Amazon S3 Glacier jobs.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Glacier;
using Amazon.Glacier.Model;

class ListJobs
{
    static async Task Main(string[] args)
    {
        var client = new AmazonGlacierClient();
        var vaultName = "example-vault";

        var request = new ListJobsRequest
        {
            // Using a hyphen "=" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the default user.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await client.ListJobsAsync(request);

        if (response.JobList.Count > 0)
        {
            response.JobList.ForEach(job => {
                Console.WriteLine($"{job.CreationDate} {job.JobDescription}");
            });
        }
        else
        {
            Console.WriteLine($"No jobs were found for {vaultName}.");
        }
    }
}
```

- For API details, see [ListJobs](#) in *AWS SDK for .NET API Reference*.

## List tags

The following code example shows how to list tags for an Amazon S3 Glacier vault.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
```

```
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Glacier;
using Amazon.Glacier.Model;

class ListTagsForVault
{
    static async Task Main(string[] args)
    {
        var client = new AmazonGlacierClient();
        var vaultName = "example-vault";

        var request = new ListTagsForVaultRequest
        {
            // Using a hyphen "=" for the Account Id will
            // cause the SDK to use the Account Id associated
            // with the default user.
            AccountId = "-",
            VaultName = vaultName,
        };

        var response = await client.ListTagsForVaultAsync(request);

        if (response.Tags.Count > 0)
        {
            foreach (KeyValuePair<string, string> tag in response.Tags)
            {
                Console.WriteLine($"Key: {tag.Key}, value: {tag.Value}");
            }
        }
        else
        {
            Console.WriteLine($"{vaultName} has no tags.");
        }
    }
}
```

- For API details, see [ListTagsForVault](#) in [AWS SDK for .NET API Reference](#).

## List vaults

The following code example shows how to list Amazon S3 Glacier vaults.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Glacier;
using Amazon.Glacier.Model;

public class ListVaults
{
    public static async Task Main(string[] args)
    {
        var client = new AmazonGlacierClient();
```

```
var request = new ListVaultsRequest
{
    AccountId = "-",
    Limit = 5,
};

var response = await client.ListVaultsAsync(request);

List<DescribeVaultOutput> vaultList = response.VaultList;

vaultList.ForEach(v => { Console.WriteLine($"{v.VaultName} ARN: {v.VaultARN}"); });
}
```

- For API details, see [ListVaults](#) in *AWS SDK for .NET API Reference*.

## Upload an archive to a vault

The following code example shows how to upload an archive to an Amazon S3 Glacier vault.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.Glacier;
using Amazon.Glacier.Transfer;

public class UploadArchiveHighLevel
{
    private static readonly string VaultName = "example-vault";
    private static readonly string ArchiveToUpload = "*** Provide file name (with
full path) to upload ***";

    public static async Task Main()
    {
        try
        {
            var manager = new ArchiveTransferManager(RegionEndpoint.USWest2);

            // Upload an archive.
            var response = await manager.UploadAsync(VaultName, "upload archive
test", ArchiveToUpload);

            Console.WriteLine("Copy and save the ID for use in other examples.");
            Console.WriteLine($"Archive ID: {response.ArchiveId}");
            Console.WriteLine("To continue, press Enter");
            Console.ReadKey();
        }
        catch (AmazonGlacierException ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}
```

- For API details, see [UploadArchive](#) in *AWS SDK for .NET API Reference*.

## Amazon SES examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon Simple Email Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2538\)](#)

## Actions

### Create an email template

The following code example shows how to create an Amazon SES email template.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Create an email template.
///</summary>
///<param name="name">Name of the template.</param>
///<param name="subject">Email subject.</param>
///<param name="text">Email body text.</param>
///<param name="html">Email HTML body text.</param>
///<returns>True if successful.</returns>
public async Task<bool> CreateEmailTemplateAsync(string name, string subject,
string text,
string html)
{
    var success = false;
    try
    {
        var response = await _amazonSimpleEmailService.CreateTemplateAsync(
            new CreateTemplateRequest
            {
                Template = new Template
                {
                    TemplateName = name,
                    SubjectPart = subject,
                    TextPart = text,
                    HtmlPart = html
                }
            });
        success = response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (Exception ex)
{
```

```
        Console.WriteLine("CreateEmailTemplateAsync failed with exception: " +
    ex.Message);
}

return success;
}
```

- For API details, see [CreateTemplate in AWS SDK for .NET API Reference](#).

## Delete an email template

The following code example shows how to delete an Amazon SES email template.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Delete an email template.
/// </summary>
/// <param name="templateName">Name of the template.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteEmailTemplateAsync(string templateName)
{
    var success = false;
    try
    {
        var response = await _amazonSimpleEmailService.DeleteTemplateAsync(
            new DeleteTemplateRequest
            {
                TemplateName = templateName
            });
        success = response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (Exception ex)
    {
        Console.WriteLine("DeleteEmailTemplateAsync failed with exception: " +
    ex.Message);
    }

    return success;
}
```

- For API details, see [DeleteTemplate in AWS SDK for .NET API Reference](#).

## Delete an identity

The following code example shows how to delete an Amazon SES identity.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Delete an email identity.
/// </summary>
/// <param name="identityEmail">The identity email to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteIdentityAsync(string identityEmail)
{
    var success = false;
    try
    {
        var response = await _amazonSimpleEmailService.DeleteIdentityAsync(
            new DeleteIdentityRequest
            {
                Identity = identityEmail
            });
        success = response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (Exception ex)
    {
        Console.WriteLine("DeleteIdentityAsync failed with exception: " +
            ex.Message);
    }

    return success;
}
```

- For API details, see [DeleteIdentity](#) in *AWS SDK for .NET API Reference*.

## Get sending limits

The following code example shows how to get Amazon SES sending limits.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get information on the current account's send quota.
/// </summary>
/// <returns>The send quota response data.</returns>
public async Task<GetSendQuotaResponse> GetSendQuotaAsync()
{
    var result = new GetSendQuotaResponse();
    try
    {
        var response = await _amazonSimpleEmailService.GetSendQuotaAsync(
            new GetSendQuotaRequest());
        result = response;
    }
    catch (Exception ex)
    {
        Console.WriteLine("GetSendQuotaAsync failed with exception: " +
            ex.Message);
    }
}
```

```
        return result;  
    }
```

- For API details, see [GetSendQuota](#) in *AWS SDK for .NET API Reference*.

## Get the status of an identity

The following code example shows how to get the status of an Amazon SES identity.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>  
/// Get identity verification status for an email.  
///</summary>  
///<returns>The verification status of the email.</returns>  
public async Task<VerificationStatus> GetIdentityStatusAsync(string email)  
{  
    var result = VerificationStatus.TemporaryFailure;  
    try  
    {  
        var response =  
            await _amazonSimpleEmailService.GetIdentityVerificationAttributesAsync(  
                new GetIdentityVerificationAttributesRequest  
                {  
                    Identities = new List<string> { email }  
                });  
  
        if (response.VerificationAttributes.ContainsKey(email))  
            result = response.VerificationAttributes[email].VerificationStatus;  
    }  
    catch (Exception ex)  
    {  
        Console.WriteLine("GetIdentityStatusAsync failed with exception: " +  
            ex.Message);  
    }  
  
    return result;  
}
```

- For API details, see [GetIdentityVerificationAttributes](#) in *AWS SDK for .NET API Reference*.

## List email templates

The following code example shows how to list Amazon SES email templates.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// List email templates for the current account.
/// </summary>
/// <returns>A list of template metadata.</returns>
public async Task<List<TemplateMetadata>> ListEmailTemplatesAsync()
{
    var result = new List<TemplateMetadata>();
    try
    {
        var response = await _amazonSimpleEmailService.ListTemplatesAsync(
            new ListTemplatesRequest());
        result = response.TemplatesMetadata;
    }
    catch (Exception ex)
    {
        Console.WriteLine("ListEmailTemplatesAsync failed with exception: " +
            ex.Message);
    }

    return result;
}
```

- For API details, see [ListTemplates](#) in *AWS SDK for .NET API Reference*.

## List identities

The following code example shows how to list Amazon SES identities.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get the identities of a specified type for the current account.
/// </summary>
/// <param name="identityType">IdentityType to list.</param>
/// <returns>The list of identities.</returns>
public async Task<List<string>> ListIdentitiesAsync(IdentityType identityType)
{
    var result = new List<string>();
    try
    {
        var response = await _amazonSimpleEmailService.ListIdentitiesAsync(
            new ListIdentitiesRequest
            {
                IdentityType = identityType
            });
        result = response.Identities;
    }
    catch (Exception ex)
    {
        Console.WriteLine("ListIdentitiesAsync failed with exception: " +
            ex.Message);
    }

    return result;
}
```

```
}
```

- For API details, see [ListIdentities](#) in [AWS SDK for .NET API Reference](#).

## Send email

The following code example shows how to send email with Amazon SES.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Send an email by using Amazon SES.
///</summary>
///<param name="toAddresses">List of recipients.</param>
///<param name="ccAddresses">List of cc recipients.</param>
///<param name="bccAddresses">List of bcc recipients.</param>
///<param name="bodyHtml">Body of the email in HTML.</param>
///<param name="bodyText">Body of the email in plain text.</param>
///<param name="subject">Subject line of the email.</param>
///<param name="senderAddress">From address.</param>
///<returns>The messageId of the email.</returns>
public async Task<string> SendEmailAsync(List<string> toAddresses,
                                         List<string> ccAddresses, List<string> bccAddresses,
                                         string bodyHtml, string bodyText, string subject, string senderAddress)
{
    var messageId = "";
    try
    {
        var response = await _amazonSimpleEmailService.SendEmailAsync(
            new SendEmailRequest
            {
                Destination = new Destination
                {
                    BccAddresses = bccAddresses,
                    CcAddresses = ccAddresses,
                    ToAddresses = toAddresses
                },
                Message = new Message
                {
                    Body = new Body
                    {
                        Html = new Content
                        {
                            Charset = "UTF-8",
                            Data = bodyHtml
                        },
                        Text = new Content
                        {
                            Charset = "UTF-8",
                            Data = bodyText
                        }
                    },
                    Subject = new Content
                    {
                        Charset = "UTF-8",

```

```
        Data = subject
    }
},
Source = senderAddress
});
messageId = response.MessageId;
}
catch (Exception ex)
{
    Console.WriteLine("SendEmailAsync failed with exception: " + ex.Message);
}

return messageId;
}
```

- For API details, see [SendEmail](#) in *AWS SDK for .NET API Reference*.

## Send templated email

The following code example shows how to send templated email with Amazon SES.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Send an email using a template.
/// </summary>
/// <param name="sender">Address of the sender.</param>
/// <param name="recipients">Addresses of the recipients.</param>
/// <param name="templateName">Name of the email template.</param>
/// <param name="templateDataObject">Data for the email template.</param>
/// <returns>The messageId of the email.</returns>
public async Task<string> SendTemplateEmailAsync(string sender, List<string>
recipients,
        string templateName, object templateDataObject)
{
    var messageId = "";
    try
    {
        // Template data should be serialized JSON from either a class or a dynamic
object.
        var templateData = JsonSerializer.Serialize(templateDataObject);

        var response = await _amazonSimpleEmailService.SendTemplatedEmailAsync(
            new SendTemplatedEmailRequest
            {
                Source = sender,
                Destination = new Destination
                {
                    ToAddresses = recipients
                },
                Template = templateName,
                TemplateData = templateData
            });
        messageId = response.MessageId;
    }
    catch (Exception ex)
```

```
{  
    Console.WriteLine("SendTemplateEmailAsync failed with exception: " +  
ex.Message);  
}  
  
    return messageId;  
}
```

- For API details, see [SendTemplatedEmail](#) in *AWS SDK for .NET API Reference*.

## Verify an email identity

The following code example shows how to verify an email identity with Amazon SES.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>  
/// Starts verification of an email identity. This request sends an email  
/// from Amazon SES to the specified email address. To complete  
/// verification, follow the instructions in the email.  
/// </summary>  
/// <param name="recipientEmailAddress">Email address to verify.</param>  
/// <returns>True if successful.</returns>  
public async Task<bool> VerifyEmailIdentityAsync(string recipientEmailAddress)  
{  
    var success = false;  
    try  
    {  
        var response = await _amazonSimpleEmailService.VerifyEmailIdentityAsync(  
            new VerifyEmailIdentityRequest  
            {  
                EmailAddress = recipientEmailAddress  
            });  
  
        success = response.HttpStatusCode == HttpStatusCode.OK;  
    }  
    catch (Exception ex)  
    {  
        Console.WriteLine("VerifyEmailIdentityAsync failed with exception: " +  
ex.Message);  
    }  
  
    return success;  
}
```

- For API details, see [VerifyEmailIdentity](#) in *AWS SDK for .NET API Reference*.

## Amazon SNS examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon Simple Notification Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2546\)](#)

## Actions

### Check whether a phone number is opted out

The following code example shows how to check whether a phone number is opted out of receiving Amazon SNS messages.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

///<summary>
/// This example shows how to use the Amazon Simple Notification Service
/// (Amazon SNS) to check whether a phone number has been opted out. The
/// example was created using the AWS SDK for .NET version 3.7 and
/// .NET Core 5.0.
///</summary>
public class IsPhoneNumOptedOut
{
    public static async Task Main()
    {
        string phoneNumber = "+15551112222";

        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        await CheckIfOptedOutAsync(client, phoneNumber);
    }

    ///<summary>
    /// Checks to see if the supplied phone number has been opted out.
    ///</summary>
    ///<param name="client">The initialized Amazon SNS Client object used
    /// to check if the phone number has been opted out.</param>
    ///<param name="phoneNumber">A string representing the phone number
    /// to check.</param>
    public static async Task CheckIfOptedOutAsync(IAmazonSimpleNotificationService
client, string phoneNumber)
    {
        var request = new CheckIfPhoneNumberIsOptedOutRequest
        {
            PhoneNumber = phoneNumber,
        };

        try
        {
            var response = await client.CheckIfPhoneNumberIsOptedOutAsync(request);
        }
    }
}
```

```
        if (response.StatusCode == System.Net.HttpStatusCode.OK)
    {
        string optOutStatus = response.IsOptedOut ? "opted out" : "not
opted out.";
        Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
    }
    catch (AuthorizationErrorException ex)
    {
        Console.WriteLine($"{ex.Message}");
    }
}
```

- For API details, see [CheckIfPhoneNumberIsOptedOut](#) in *AWS SDK for .NET API Reference*.

## Create a topic

The following code example shows how to create an Amazon SNS topic.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example shows how to use Amazon Simple Notification Service
/// (Amazon SNS) to add a new Amazon SNS topic. The example was created
/// using the AWS SDK for .NET version 3.7 and .NET Core 5.0.
/// </summary>
public class CreateSNSTopic
{
    public static async Task Main()
    {
        string topicName = "ExampleSNSTopic";

        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        var topicArn = await CreateSNSTopicAsync(client, topicName);
        Console.WriteLine($"New topic ARN: {topicArn}");
    }

    /// <summary>
    /// Creates a new SNS topic using the supplied topic name.
    /// </summary>
    /// <param name="client">The initialized SNS client object used to
    /// create the new topic.</param>
    /// <param name="topicName">A string representing the topic name.</param>
    /// <returns>The Amazon Resource Name (ARN) of the created topic.</returns>
    public static async Task<string>
CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
    {
```

```
        var request = new CreateTopicRequest
        {
            Name = topicName,
        };

        var response = await client.CreateTopicAsync(request);

        return response.TopicArn;
    }

}
```

- For API details, see [CreateTopic](#) in *AWS SDK for .NET API Reference*.

## Delete a subscription

The following code example shows how to delete an Amazon SNS subscription.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Given the ARN for an Amazon SNS subscription, this method deletes
/// the subscription.
/// </summary>
/// <param name="client">The initialized Amazon SNS client object, used
/// to delete an Amazon SNS subscription.</param>
/// <param name="subscriptionArn">The ARN of the subscription to delete.</
param>
public static async Task TopicUnsubscribeAsync(
    IAmazonSimpleNotificationService client,
    string subscriptionArn)
{
    var response = await client.UnsubscribeAsync(subscriptionArn);
}
```

- For API details, see [Unsubscribe](#) in *AWS SDK for .NET API Reference*.

## Delete a topic

The following code example shows how to delete an Amazon SNS topic and all subscriptions to that topic.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
```

```
/// <summary>
/// This example deletes an existing Amazon Simple Notification Service
/// (Amazon SNS) topic. The example was created using the AWS SDK for .NET
/// version 3.7 and .NET Core 5.0.
/// </summary>
public class DeleteSNSTopic
{
    public static async Task Main()
    {
        string topicArn = "arn:aws:sns:us-east-2:012345678901:ExampleSNSTopic";
        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        var response = await client.DeleteTopicAsync(topicArn);
    }
}
```

- For API details, see [DeleteTopic](#) in *AWS SDK for .NET API Reference*.

## Get the properties of a topic

The following code example shows how to get the properties of an Amazon SNS topic.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example shows how to retrieve the attributes of an Amazon Simple
/// Notification Service (Amazon SNS) topic. The example was written using
/// the AWS SDK for .NET 3.7 and .NET Core 5.0.
/// </summary>
public class GetTopicAttributes
{
    public static async Task Main()
    {
        string topicArn = "arn:aws:sns:us-west-2:000000000000:ExampleSNSTopic";
        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        var attributes = await GetTopicAttributesAsync(client, topicArn);
        DisplayTopicAttributes(attributes);
    }

    /// <summary>
    /// Given the ARN of the Amazon SNS topic, this method retrieves the topic
    /// attributes.
    /// </summary>
    /// <param name="client">The initialized Amazon SNS client object used
    /// to retrieve the attributes for the Amazon SNS topic.</param>
    /// <param name="topicArn">The ARN of the topic for which to retrieve
    /// the attributes.</param>
}
```

```
/// <returns>A Dictionary of topic attributes.</returns>
public static async Task<Dictionary<string, string>> GetTopicAttributesAsync(
    IAmazonSimpleNotificationService client,
    string topicArn)
{
    var response = await client.GetTopicAttributesAsync(topicArn);

    return response.Attributes;
}

/// <summary>
/// This method displays the attributes for an Amazon SNS topic.
/// </summary>
/// <param name="topicAttributes">A Dictionary containing the
/// attributes for an Amazon SNS topic.</param>
public static void DisplayTopicAttributes(Dictionary<string, string>
topicAttributes)
{
    foreach (KeyValuePair<string, string> entry in topicAttributes)
    {
        Console.WriteLine($"{entry.Key}: {entry.Value}\n");
    }
}
```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for .NET API Reference*.

### List the subscribers of a topic

The following code example shows how to retrieve the list of subscribers of an Amazon SNS topic.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example will retrieve a list of the existing Amazon Simple
/// Notification Service (Amazon SNS) subscriptions. The example was
/// created using the AWS SDK for .NET 3.7 and .NET Core 5.0.
/// </summary>
public class ListSubscriptions
{
    public static async Task Main()
    {
        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        var subscriptions = await GetSubscriptionsListAsync(client);

        DisplaySubscriptionList(subscriptions);
    }

    /// <summary>
```

```
    /// Gets a list of the existing Amazon SNS subscriptions.  
    /// </summary>  
    /// <param name="client">The initialized Amazon SNS client object used  
    /// to obtain the list of subscriptions.</param>  
    /// <returns>A List containing information about each subscription.</returns>  
    public static async Task<List<Subscription>>  
        GetSubscriptionsListAsync(IAmazonSimpleNotificationService client)  
    {  
        var response = await client.ListSubscriptionsAsync();  
  
        return response.Subscriptions;  
    }  
  
    /// <summary>  
    /// Display a list of Amazon SNS subscription information.  
    /// </summary>  
    /// <param name="subscriptionList">A list containing details for existing  
    /// Amazon SNS subscriptions.</param>  
    public static void DisplaySubscriptionList(List<Subscription> subscriptionList)  
    {  
        foreach (var subscription in subscriptionList)  
        {  
            Console.WriteLine($"Owner: {subscription.Owner}");  
            Console.WriteLine($"Subscription ARN: {subscription.SubscriptionArn}");  
            Console.WriteLine($"Topic ARN: {subscription.TopicArn}");  
            Console.WriteLine($"Endpoint: {subscription.Endpoint}");  
            Console.WriteLine($"Protocol: {subscription.Protocol}");  
            Console.WriteLine();  
        }  
    }  
}
```

- For API details, see [ListSubscriptions](#) in *AWS SDK for .NET API Reference*.

## List topics

The following code example shows how to list Amazon SNS topics.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Collections.Generic;  
using System.Threading.Tasks;  
using Amazon.SimpleNotificationService;  
using Amazon.SimpleNotificationService.Model;  
  
    /// <summary>  
    /// An example to list the Amazon Simple Notification Service (Amazon SNS)  
    /// topics for the default user account. The code was written using the  
    /// AWS SDK for .NET 3.7 and .NET Core 5.0.  
    /// </summary>  
    public class ListSNSTopics  
    {  
        public static async Task Main()  
        {  
            IAmazonSimpleNotificationService client = new  
                AmazonSimpleNotificationServiceClient();
```

```
        await GetTopicListAsync(client);
    }

    /// <summary>
    /// Retrieves the list of Amazon SNS topics in groups of up to 100
    /// topics.
    /// </summary>
    /// <param name="client">The initialized Amazon SNS client object used
    /// to retrieve the list of topics.</param>
    public static async Task GetTopicListAsync(IAmazonSimpleNotificationService
client)
{
    // If there are more than 100 Amazon SNS topics, the call to
    // ListTopicsAsync will return a value to pass to the
    // method to retrieve the next 100 (or less) topics.
    string nextToken = string.Empty;

    do
    {
        var response = await client.ListTopicsAsync(nextToken);
        DisplayTopicsList(response.Topics);
        nextToken = response.NextToken;
    }
    while (!string.IsNullOrEmpty(nextToken));
}

    /// <summary>
    /// Displays the list of Amazon SNS Topic ARNs.
    /// </summary>
    /// <param name="topicList">The list of Topic ARNs.</param>
    public static void DisplayTopicsList(List<Topic> topicList)
{
    foreach (var topic in topicList)
    {
        Console.WriteLine($"{topic.TopicArn}");
    }
}
}
```

- For API details, see [ListTopics](#) in *AWS SDK for .NET API Reference*.

## Publish an SMS text message

The following code example shows how to publish SMS messages using Amazon SNS.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;
using System;
using System.Threading.Tasks;

namespace SNSMessageExample
{
    class SNSMessage
```

```
{  
    private AmazonSimpleNotificationServiceClient snsClient;  
  
    /// <summary>  
    /// Constructs a new SNSMessage object initializing the Amazon Simple  
    /// Notification Service (Amazon SNS) client using the supplied  
    /// Region endpoint.  
    /// </summary>  
    /// <param name="regionEndpoint">The Amazon Region endpoint to use in  
    /// sending test messages with this object.</param>  
    public SNSMessage(RegionEndpoint regionEndpoint)  
    {  
        snsClient = new AmazonSimpleNotificationServiceClient(regionEndpoint);  
    }  
  
    /// <summary>  
    /// Sends the SMS message passed in the text parameter to the phone number  
    /// in phoneNum.  
    /// </summary>  
    /// <param name="phoneNum">The ten-digit phone number to which the text  
    /// message will be sent.</param>  
    /// <param name="text">The text of the message to send.</param>  
    /// <returns></returns>  
    public async Task SendTextMessageAsync(string phoneNum, string text)  
    {  
        if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))  
        {  
            return;  
        }  
  
        // Now actually send the message.  
        var request = new PublishRequest  
        {  
            Message = text,  
            PhoneNumber = phoneNum  
        };  
  
        try  
        {  
            var response = await snsClient.PublishAsync(request);  
        }  
        catch (Exception ex)  
        {  
            Console.WriteLine($"Error sending message: {ex}");  
        }  
    }  
}
```

- For API details, see [Publish](#) in [AWS SDK for .NET API Reference](#).

## Publish to a topic

The following code example shows how to publish messages to an Amazon SNS topic.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

- For API details, see [Publish](#) in [AWS SDK for .NET API Reference](#).

## Subscribe an email address to a topic

The following code example shows how to subscribe an email address to an Amazon SNS topic.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Creates a new subscription to a topic.
/// </summary>
/// <param name="client">The initialized Amazon SNS client object, used
/// to create an Amazon SNS subscription.</param>
/// <param name="topicArn">The ARN of the topic to subscribe to.</param>
/// <returns>A SubscribeResponse object which includes the subscription
/// ARN for the new subscription.</returns>
public static async Task<SubscribeResponse> TopicSubscribeAsync(
    IAmazonSimpleNotificationService client,
    string topicArn)
{
    SubscribeRequest request = new SubscribeRequest()
    {
        TopicArn = topicArn,
        ReturnSubscriptionArn = true,
        Protocol = "email",
        Endpoint = "recipient@example.com",
    };
    var response = await client.SubscribeAsync(request);
    return response;
}
```

- For API details, see [Subscribe](#) in *AWS SDK for .NET API Reference*.

## Amazon SQS examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon Simple Queue Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2554\)](#)

## Actions

### Authorize a bucket to send messages to a queue

The following code example shows how to authorize an Amazon S3 bucket to send messages to an Amazon SQS queue.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.SQS;

public class AuthorizeS3ToSendMessage
{
    /// <summary>
    /// Initializes the Amazon SQS client object and then calls the
    /// AuthorizeS3ToSendMessageAsync method to authorize the named
    /// bucket to send messages in response to S3 events.
    /// </summary>
    public static async Task Main()
    {
        string queueUrl = "https://sqs.us-east-2.amazonaws.com/0123456789ab/
Example_Queue";
        string bucketName = "doc-example-bucket";

        // Create an Amazon SQS client object using the
        // default user. If the AWS Region you want to use
        // is different, supply the AWS Region as a parameter.
        IAmazonSQS client = new AmazonSQSClient();

        var queueARN = await client.AuthorizeS3ToSendMessageAsync(queueUrl,
bucketName);

        if (!string.IsNullOrEmpty(queueARN))
        {
            Console.WriteLine($"The Amazon S3 bucket: {bucketName} has been
successfully authorized.");
            Console.WriteLine($"{bucketName} can now send messages to the queue
with ARN: {queueARN}.");
        }
    }
}
```

## Create a queue

The following code example shows how to create an Amazon SQS queue.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SQS;
using Amazon.SQS.Model;

public class CreateQueue
```

```
{
    /// <summary>
    /// Initializes the Amazon SQS client object and then calls the
    /// CreateQueueAsync method to create the new queue. If the call is
    /// successful, it displays the URL of the new queue on the console.
    /// </summary>
    public static async Task Main()
    {
        // If the Amazon SQS message queue is not in the same AWS Region as your
        // default user, you need to provide the AWS Region as a parameter to the
        // client constructor.
        var client = new AmazonSQSClient();

        string queueName = "New-Example-Queue";
        int maxMessage = 256 * 1024;
        var attrs = new Dictionary<string, string>
        {
            {
                QueueAttributeName.DelaySeconds,
                TimeSpan.FromSeconds(5).TotalSeconds.ToString()
            },
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            },
            {
                QueueAttributeName.MessageRetentionPeriod,
                TimeSpan.FromDays(4).TotalSeconds.ToString()
            },
            {
                QueueAttributeName.ReceiveMessageWaitTimeSeconds,
                TimeSpan.FromSeconds(5).TotalSeconds.ToString()
            },
            {
                QueueAttributeName.VisibilityTimeout,
                TimeSpan.FromHours(12).TotalSeconds.ToString()
            },
        };
        var request = new CreateQueueRequest
        {
            Attributes = attrs,
            QueueName = queueName,
        };

        var response = await client.CreateQueueAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("Successfully created Amazon SQS queue.");
            Console.WriteLine($"Queue URL: {response.QueueUrl}");
        }
    }
}
```

Create an Amazon SQS queue and send a message to it.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.SQS;
using Amazon.SQS.Model;
```

```
public class CreateSendExample
{
    // Specify your AWS Region (an example Region is shown).
    private static readonly string QueueName = "Example_Queue";
    private static readonly RegionEndpoint ServiceRegion = RegionEndpoint.USWest2;
    private static IAmazonSQS client;

    public static async Task Main()
    {
        client = new AmazonSQSClient(ServiceRegion);
        var createQueueResponse = await CreateQueue(client, QueueName);

        string queueUrl = createQueueResponse.QueueUrl;

        Dictionary<string, MessageAttributeValue> messageAttributes = new
        Dictionary<string, MessageAttributeValue>
        {
            { "Title", new MessageAttributeValue { DataType = "String",
StringValue = "The Whistler" } },
            { "Author", new MessageAttributeValue { DataType = "String",
StringValue = "John Grisham" } },
            { "WeeksOn", new MessageAttributeValue { DataType = "Number",
StringValue = "6" } };
        };

        string messageBody = "Information about current NY Times fiction bestseller
for week of 12/11/2016./";

        var sendMsgResponse = await SendMessage(client, queueUrl, messageBody,
messageAttributes);
    }

    ///<summary>
    /// Creates a new Amazon SQS queue using the queue name passed to it
    /// in queueName.
    ///</summary>
    ///<param name="client">An SQS client object used to send the message.</param>
    ///<param name="queueName">A string representing the name of the queue
    /// to create.</param>
    ///<returns>A CreateQueueResponse that contains information about the
    /// newly created queue.</returns>
    public static async Task<CreateQueueResponse> CreateQueue(IAmazonSQS client,
string queueName)
    {
        var request = new CreateQueueRequest
        {
            QueueName = queueName,
            Attributes = new Dictionary<string, string>
            {
                { "DelaySeconds", "60" },
                { "MessageRetentionPeriod", "86400" },
            },
        };

        var response = await client.CreateQueueAsync(request);
        Console.WriteLine($"Created a queue with URL : {response.QueueUrl}");

        return response;
    }

    ///<summary>
    /// Sends a message to an SQS queue.
    ///</summary>
    ///<param name="client">An SQS client object used to send the message.</param>
    ///<param name="queueUrl">The URL of the queue to which to send the

```

```
/// message.</param>
/// <param name="messageBody">A string representing the body of the
/// message to be sent to the queue.</param>
/// <param name="messageAttributes">Attributes for the message to be
/// sent to the queue.</param>
/// <returns>A SendMessageResponse object that contains information
/// about the message that was sent.</returns>
public static async Task<SendMessageResponse> SendMessage(
    IAmazonSQS client,
    string queueUrl,
    string messageBody,
    Dictionary<string, MessageAttributeValue> messageAttributes)
{
    var sendMessageRequest = new SendMessageRequest
    {
        DelaySeconds = 10,
        MessageAttributes = messageAttributes,
        MessageBody = messageBody,
        QueueUrl = queueUrl,
    };

    var response = await client.SendMessageAsync(sendMessageRequest);
    Console.WriteLine($"Sent a message with id : {response.MessageId}");

    return response;
}
```

- For API details, see [CreateQueue](#) in *AWS SDK for .NET API Reference*.

## Delete a message from a queue

The following code example shows how to delete a message from an Amazon SQS queue.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SQS;
using Amazon.SQS.Model;

public class DeleteMessage
{
    /// <summary>
    /// Initializes the Amazon SQS client object. It then calls the
    /// ReceiveMessageAsync method to retrieve information about the
    /// available methods before deleting them.
    /// </summary>
    public static async Task Main()
    {
        string queueUrl = "https://sqs.us-east-2.amazonaws.com/0123456789ab/
Example_Queue";
        var attributeNames = new List<string>() { "All" };
        int maxNumberOfMessages = 5;
        var visibilityTimeout = (int)TimeSpan.FromMinutes(10).TotalSeconds;
```

```
var waitTimeSeconds = (int)TimeSpan.FromSeconds(5).TotalSeconds;

// If the Amazon SQS message queue is not in the same AWS Region as your
// default user, you need to provide the AWS Region as a parameter to the
// client constructor.
var client = new AmazonSQSClient();

var request = new ReceiveMessageRequest
{
    QueueUrl = queueUrl,
    AttributeNames = attributeNames,
    MaxNumberOfMessages = maxNumberOfMessages,
    VisibilityTimeout = visibilityTimeout,
    WaitTimeSeconds = waitTimeSeconds,
};

var response = await client.ReceiveMessageAsync(request);

if (response.Messages.Count > 0)
{
    response.Messages.ForEach(async m =>
    {
        Console.WriteLine($"Message ID: '{m.MessageId}'");

        var delRequest = new DeleteMessageRequest
        {
            QueueUrl = "https://sqs.us-east-1.amazonaws.com/0123456789ab/",
            ReceiptHandle = m.ReceiptHandle,
        };

        var delResponse = await client.DeleteMessageAsync(delRequest);
    });
}
else
{
    Console.WriteLine("No messages to delete.");
}
}
```

Receive a message from an Amazon SQS queue and then delete the message.

```
public static async Task Main()
{
    // If the AWS Region you want to use is different from
    // the AWS Region defined for the default user, supply
    // the specify your AWS Region to the client constructor.
    var client = new AmazonSQSClient();
    string queueName = "Example_Queue";

    var queueUrl = await GetQueueUrl(client, queueName);
    Console.WriteLine($"The SQS queue's URL is {queueUrl}");

    var response = await ReceiveAndDeleteMessage(client, queueUrl);

    Console.WriteLine($"Message: {response.Messages[0]}");
}

/// <summary>
/// Retrieve the queue URL for the queue named in the queueName
/// property using the client object.
/// </summary>
```

```

    ///> The Amazon SQS client used to retrieve the
    ///> queue URL.</param>
    ///> A string representing name of the queue
    ///> for which to retrieve the URL.</param>
    ///> The URL of the queue.</returns>
    public static async Task<string> GetQueueUrl(IAmazonSQS client, string
queueName)
    {
        var request = new GetQueueUrlRequest
        {
            QueueName = queueName,
        };

        GetQueueUrlResponse response = await client.GetQueueUrlAsync(request);
        return response.QueueUrl;
    }

    ///> Summary
    ///> Retrieves the message from the queue at the URL passed in the
    ///> queueURL parameters using the client.
    ///> /summary>
    ///> The SQS client used to retrieve a message.</param>
    ///> The URL of the queue from which to retrieve
    ///> a message.</param>
    ///> The response from the call to ReceiveMessageAsync.</returns>
    public static async Task<ReceiveMessageResponse>
ReceiveAndDeleteMessage(IAmazonSQS client, string queueUrl)
{
    // Receive a single message from the queue.
    var receiveMessageRequest = new ReceiveMessageRequest
    {
        AttributeNames = { "SentTimestamp" },
        MaxNumberOfMessages = 1,
        MessageAttributeNames = { "All" },
        QueueUrl = queueUrl,
        VisibilityTimeout = 0,
        WaitTimeSeconds = 0,
    };

    var receiveMessageResponse = await
client.ReceiveMessageAsync(receiveMessageRequest);

    // Delete the received message from the queue.
    var deleteMessageRequest = new DeleteMessageRequest
    {
        QueueUrl = queueUrl,
        ReceiptHandle = receiveMessageResponse.Messages[0].ReceiptHandle,
    };

    await client.DeleteMessageAsync(deleteMessageRequest);

    return receiveMessageResponse;
}
}

```

- For API details, see [DeleteMessage](#) in *AWS SDK for .NET API Reference*.

## Delete a queue

The following code example shows how to delete an Amazon SQS queue.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.SQS;

public class DeleteQueue
{
    /// <summary>
    /// Initializes the Amazon SQS client object and then calls the
    /// DeleteQueueAsync method to delete the queue.
    /// </summary>
    public static async Task Main()
    {
        // If the Amazon SQS message queue is not in the same AWS Region as your
        // default user, you need to provide the AWS Region as a parameter to the
        // client constructor.
        var client = new AmazonSQSClient();

        string queueUrl = "https://sqs.us-east-2.amazonaws.com/0123456789ab/New-
Example-Queue";

        var response = await client.DeleteQueueAsync(queueUrl);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("Successfully deleted the queue.");
        }
        else
        {
            Console.WriteLine("Could not delete the queue.");
        }
    }
}
```

- For API details, see [DeleteQueue](#) in *AWS SDK for .NET API Reference*.

## Get attributes for a queue

The following code example shows how to get attributes for an Amazon SQS queue.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SQS;
using Amazon.SQS.Model;

public class GetQueueAttributes
```

```
{
    /// <summary>
    /// Initializes the Amazon SQS client and then uses it to call the
    /// GetQueueAttributesAsync method to retrieve the attributes for the
    /// Amazon SQS queue.
    /// </summary>
    public static async Task Main()
    {
        // If the Amazon SQS message queue is not in the same AWS Region as your
        // default user, you need to provide the AWS Region as a parameter to the
        // client constructor.
        var client = new AmazonSQSClient();

        var queueUrl = "https://sqs.us-east-2.amazonaws.com/0123456789ab/New-
Example-Queue";
        var attrs = new List<string>() { "All" };

        var request = new GetQueueAttributesRequest
        {
            QueueUrl = queueUrl,
            AttributeNames = attrs,
        };

        var response = await client.GetQueueAttributesAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            DisplayAttributes(response);
        }
    }

    /// <summary>
    /// Displays the attributes passed to the method on the console.
    /// </summary>
    /// <param name="attrs">The attributes for the Amazon SQS queue.</param>
    public static void DisplayAttributes(GetQueueAttributesResponse attrs)
    {
        Console.WriteLine($"Attributes for queue ARN '{attrs.QueueARN}':");
        Console.WriteLine($" Approximate number of messages:
{attrs.ApproximateNumberOfMessages}");
        Console.WriteLine($" Approximate number of messages delayed:
{attrs.ApproximateNumberOfMessagesDelayed}");
        Console.WriteLine($" Approximate number of messages not visible:
{attrs.ApproximateNumberOfMessagesNotVisible}");
        Console.WriteLine($" Queue created on: {attrs.CreatedTimestamp}");
        Console.WriteLine($" Delay seconds: {attrs.DelaySeconds}");
        Console.WriteLine($" Queue last modified on:
{attrs.LastModifiedTimestamp}");
        Console.WriteLine($" Maximum message size: {attrs.MaximumMessageSize}");
        Console.WriteLine($" Message retention period:
{attrs.MessageRetentionPeriod}");
        Console.WriteLine($" Visibility timeout: {attrs.VisibilityTimeout}");
        Console.WriteLine($" Policy: {attrs.Policy}\n");
        Console.WriteLine(" Attributes:");

        foreach (var attr in attrs.Attributes)
        {
            Console.WriteLine($"      {attr.Key}: {attr.Value}");
        }
    }
}
```

- For API details, see [GetQueueAttributes](#) in *AWS SDK for .NET API Reference*.

## Get the URL of a queue

The following code example shows how to get the URL of an Amazon SQS queue.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.SQS;
using Amazon.SQS.Model;

public class GetQueueUrl
{
    /// <summary>
    /// Initializes the Amazon SQS client object and then calls the
    /// GetQueueUrlAsync method to retrieve the URL of an Amazon SQS
    /// queue.
    /// </summary>
    public static async Task Main()
    {
        // If the Amazon SQS message queue is not in the same AWS Region as your
        // default user, you need to provide the AWS Region as a parameter to the
        // client constructor.
        var client = new AmazonSQSClient();

        string queueName = "New-Exampe-Queue";

        try
        {
            var response = await client.GetQueueUrlAsync(queueName);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                Console.WriteLine($"The URL for {queueName} is:
{response.QueueUrl}");
            }
        }
        catch (QueueDoesNotExistException ex)
        {
            Console.WriteLine(ex.Message);
            Console.WriteLine($"The queue {queueName} was not found.");
        }
    }
}
```

- For API details, see [GetQueueUrl](#) in *AWS SDK for .NET API Reference*.

## Receive messages from a queue

The following code example shows how to receive messages from an Amazon SQS queue.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SQS;
using Amazon.SQS.Model;

public class ReceiveFromQueue
{
    public static async Task Main(string[] args)
    {
        string queueUrl = "https://sqs.us-east-2.amazonaws.com/0123456789ab/
Example_Queue";
        var attributeNames = new List<string>() { "All" };
        int maxNumberOfMessages = 5;
        var visibilityTimeout = (int)TimeSpan.FromMinutes(10).TotalSeconds;
        var waitTimeSeconds = (int)TimeSpan.FromSeconds(5).TotalSeconds;

        // If the Amazon SQS message queue is not in the same AWS Region as your
        // default user, you need to provide the AWS Region as a parameter to the
        // client constructor.
        var client = new AmazonSQSClient();

        var request = new ReceiveMessageRequest
        {
            QueueUrl = queueUrl,
            AttributeNames = attributeNames,
            MaxNumberOfMessages = maxNumberOfMessages,
            VisibilityTimeout = visibilityTimeout,
            WaitTimeSeconds = waitTimeSeconds,
        };

        var response = await client.ReceiveMessageAsync(request);

        if (response.Messages.Count > 0)
        {
            DisplayMessages(response.Messages);
        }
    }

    ///<summary>
    ///<summary>Display message information for a list of Amazon SQS messages.
    ///</summary>
    ///<param name="messages">The list of Amazon SQS Message objects to display.</
param>
    public static void DisplayMessages(List<Message> messages)
    {
        messages.ForEach(m =>
        {
            Console.WriteLine($"For message ID {m.MessageId}:");
            Console.WriteLine($"  Body: {m.Body}");
            Console.WriteLine($"  Receipt handle: {m.ReceiptHandle}");
            Console.WriteLine($"  MD5 of body: {m.MD5OfBody}");
            Console.WriteLine($"  MD5 of message attributes:
{m.MD5OfMessageAttributes}");
            Console.WriteLine("  Attributes:");

            foreach (var attr in m.Attributes)
            {
                Console.WriteLine($"{attr.Key}: {attr.Value}");
            }
        });
    }
}
```

Receive a message from an Amazon SQS queue, and then delete the message.

```
public static async Task Main()
{
    // If the AWS Region you want to use is different from
    // the AWS Region defined for the default user, supply
    // the specify your AWS Region to the client constructor.
    var client = new AmazonSQSClient();
    string queueName = "Example_Queue";

    var queueUrl = await GetQueueUrl(client, queueName);
    Console.WriteLine($"The SQS queue's URL is {queueUrl}");

    var response = await ReceiveAndDeleteMessage(client, queueUrl);

    Console.WriteLine($"Message: {response.Messages[0]}");
}

/// <summary>
/// Retrieve the queue URL for the queue named in the queueName
/// property using the client object.
/// </summary>
/// <param name="client">The Amazon SQS client used to retrieve the
/// queue URL.</param>
/// <param name="queueName">A string representing name of the queue
/// for which to retrieve the URL.</param>
/// <returns>The URL of the queue.</returns>
public static async Task<string> GetQueueUrl(IAmazonSQS client, string
queueName)
{
    var request = new GetQueueUrlRequest
    {
        QueueName = queueName,
    };

    GetQueueUrlResponse response = await client.GetQueueUrlAsync(request);
    return response.QueueUrl;
}

/// <summary>
/// Retrieves the message from the queque at the URL passed in the
/// queueURL parameters using the client.
/// </summary>
/// <param name="client">The SQS client used to retrieve a message.</param>
/// <param name="queueUrl">The URL of the queue from which to retrieve
/// a message.</param>
/// <returns>The response from the call to ReceiveMessageAsync.</returns>
public static async Task<ReceiveMessageResponse>
ReceiveAndDeleteMessage(IAmazonSQS client, string queueUrl)
{
    // Receive a single message from the queue.
    var receiveMessageRequest = new ReceiveMessageRequest
    {
        AttributeNames = { "SentTimestamp" },
        MaxNumberOfMessages = 1,
        MessageAttributeNames = { "All" },
        QueueUrl = queueUrl,
        VisibilityTimeout = 0,
        WaitTimeSeconds = 0,
    };

    var receiveMessageResponse = await
    client.ReceiveMessageAsync(receiveMessageRequest);
```

```
// Delete the received message from the queue.  
var deleteMessageRequest = new DeleteMessageRequest  
{  
    QueueUrl = queueUrl,  
    ReceiptHandle = receiveMessageResponse.Messages[0].ReceiptHandle,  
};  
  
await client.DeleteMessageAsync(deleteMessageRequest);  
  
return receiveMessageResponse;  
}  
}
```

- For API details, see [ReceiveMessage in AWS SDK for .NET API Reference](#).

## Send a message to a queue

The following code example shows how to send a message to an Amazon SQS queue.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.Threading.Tasks;  
using Amazon.SQS;  
using Amazon.SQS.Model;  
  
public class SendMessageToQueue  
{  
    ///<summary>  
    /// Initialize the Amazon SQS client object and use the  
    /// SendMessageAsync method to send a message to an Amazon SQS queue.  
    /// </summary>  
    public static async Task Main()  
    {  
        string messageBody = "This is a sample message to send to the example  
queue.";  
        string queueUrl = "https://sqs.us-east-2.amazonaws.com/0123456789ab/  
Example_Queue";  
  
        // Create an Amazon SQS client object using the  
        // default user. If the AWS Region you want to use  
        // is different, supply the AWS Region as a parameter.  
        IAmazonSQS client = new AmazonSQSClient();  
  
        var request = new SendMessageRequest  
        {  
            MessageBody = messageBody,  
            QueueUrl = queueUrl,  
        };  
  
        var response = await client.SendMessageAsync(request);  
  
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)  
        {  
            Console.WriteLine($"Successfully sent message. Message ID:  
{response.MessageId}");  
        }  
    }  
}
```

```
        }
    else
    {
        Console.WriteLine("Could not send message.");
    }
}
```

Create an Amazon SQS queue and send a message to it.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.SQS;
using Amazon.SQS.Model;

public class CreateSendExample
{
    // Specify your AWS Region (an example Region is shown).
    private static readonly string QueueName = "Example_Queue";
    private static readonly RegionEndpoint ServiceRegion = RegionEndpoint.USWest2;
    private static IAmazonSQS client;

    public static async Task Main()
    {
        client = new AmazonSQSClient(ServiceRegion);
        var createQueueResponse = await CreateQueue(client, QueueName);

        string queueUrl = createQueueResponse.QueueUrl;

        Dictionary<string, MessageAttributeValue> messageAttributes = new
Dictionary<string, MessageAttributeValue>
        {
            { "Title", new MessageAttributeValue { DataType = "String",
StringValue = "The Whistler" } },
            { "Author", new MessageAttributeValue { DataType = "String",
StringValue = "John Grisham" } },
            { "WeeksOn", new MessageAttributeValue { DataType = "Number",
StringValue = "6" } },
        };

        string messageBody = "Information about current NY Times fiction bestseller
for week of 12/11/2016.';

        var sendMsgResponse = await SendMessage(client, queueUrl, messageBody,
messageAttributes);
    }

    ///<summary>
    /// Creates a new Amazon SQS queue using the queue name passed to it
    /// in queueName.
    ///</summary>
    ///<param name="client">An SQS client object used to send the message.</param>
    ///<param name="queueName">A string representing the name of the queue
    /// to create.</param>
    ///<returns>A CreateQueueResponse that contains information about the
    /// newly created queue.</returns>
    public static async Task<CreateQueueResponse> CreateQueue(IAmazonSQS client,
string queueName)
    {
        var request = new CreateQueueRequest
        {
```

```
QueueName = queueName,
Attributes = new Dictionary<string, string>
{
    { "DelaySeconds", "60" },
    { "MessageRetentionPeriod", "86400" },
},
};

var response = await client.CreateQueueAsync(request);
Console.WriteLine($"Created a queue with URL : {response.QueueUrl}");

return response;
}

/// <summary>
/// Sends a message to an SQS queue.
/// </summary>
/// <param name="client">An SQS client object used to send the message.</param>
/// <param name="queueUrl">The URL of the queue to which to send the
/// message.</param>
/// <param name="messageBody">A string representing the body of the
/// message to be sent to the queue.</param>
/// <param name="messageAttributes">Attributes for the message to be
/// sent to the queue.</param>
/// <returns>A SendMessageResponse object that contains information
/// about the message that was sent.</returns>
public static async Task<SendMessageResponse> SendMessage(
    IAmazonSQS client,
    string queueUrl,
    string messageBody,
    Dictionary<string, MessageAttributeValue> messageAttributes)
{
    var sendMessageRequest = new SendMessageRequest
    {
        DelaySeconds = 10,
        MessageAttributes = messageAttributes,
        MessageBody = messageBody,
        QueueUrl = queueUrl,
    };

    var response = await client.SendMessageAsync(sendMessageRequest);
    Console.WriteLine($"Sent a message with id : {response.MessageId}");

    return response;
}
}
```

- For API details, see [SendMessage](#) in *AWS SDK for .NET API Reference*.

## AWS STS examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with AWS Security Token Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2569\)](#)

## Actions

### Assume a role

The following code example shows how to assume a role with AWS STS.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Threading.Tasks;

namespace AssumeRoleExample
{
    class AssumeRole
    {
        ///<summary>
        /// This example shows how to use the AWS Security Token
        /// Service (AWS STS) to assume an IAM role.
        ///
        /// NOTE: It is important that the role that will be assumed has a
        /// trust relationship with the account that will assume the role.
        ///
        /// Before you run the example, you need to create the role you want to
        /// assume and have it trust the IAM account that will assume that role.
        ///
        /// See https://docs.aws.amazon.com/IAM/latest/UserGuide/id\_roles\_create.html
        /// for help in working with roles.
        /// </summary>

        private static readonly RegionEndpoint REGION = RegionEndpoint.USWest2;

        static async Task Main()
        {
            // Create the SecurityToken client and then display the identity of the
            // default user.
            var roleArnToAssume = "arn:aws:iam::123456789012:role/testAssumeRole";

            var client = new
                Amazon.SecurityToken.AmazonSecurityTokenServiceClient(REGION);

            // Get and display the information about the identity of the default user.
            var callerIdRequest = new GetCallerIdentityRequest();
            var caller = await client.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"Original Caller: {caller.ArN}");

            // Create the request to use with the AssumeRoleAsync call.
            var assumeRoleReq = new AssumeRoleRequest()
            {
                DurationSeconds = 1600,
                RoleSessionName = "Session1",
                RoleArn = roleArnToAssume
            };

            var assumeRoleRes = await client.AssumeRoleAsync(assumeRoleReq);

            // Now create a new client based on the credentials of the caller assuming
            the role.
        }
    }
}
```

```
        var client2 = new AmazonSecurityTokenServiceClient(credentials:  
assumeRoleRes.Credentials);  
  
        // Get and display information about the caller that has assumed the  
defined role.  
        var caller2 = await client2.GetCallerIdentityAsync(callerIdRequest);  
        Console.WriteLine($"AssumedRole Caller: {caller2.ArN}");  
    }  
}  
}
```

- For API details, see [AssumeRole](#) in [AWS SDK for .NET API Reference](#).

## Secrets Manager examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with AWS Secrets Manager.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2570\)](#)

## Actions

### Get a secret value

The following code example shows how to get a Secrets Manager secret value.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;  
using System.IO;  
using System.Threading.Tasks;  
using Amazon.SecretsManager;  
using Amazon.SecretsManager.Model;  
  
/// <summary>  
/// This example uses the Amazon Web Service Secrets Manager to retrieve  
/// the secret value for the provided secret name. This example was created  
/// using the AWS SDK for .NET v3.7 and .NET Core 5.0.  
/// </summary>  
public class GetSecretValue  
{  
    /// <summary>  
    /// The main method initializes the necessary values and then calls  
    /// the GetSecretAsync and DecodeString methods to get the decoded  
    /// secret value for the secret named in secretName.  
    /// </summary>  
    public static async Task Main()
```

```
{  
    string secretName = "<<{{MySecretName}}>>";  
    string secret;  
  
    IAmazonSecretsManager client = new AmazonSecretsManagerClient();  
  
    var response = await GetSecretAsync(client, secretName);  
  
    if (response is not null)  
    {  
        secret = DecodeString(response);  
  
        if (!string.IsNullOrEmpty(secret))  
        {  
            Console.WriteLine($"The decoded secret value is: {secret}.");  
        }  
        else  
        {  
            Console.WriteLine("No secret value was returned.");  
        }  
    }  
}  
  
/// <summary>  
/// Retrieves the secret value given the name of the secret to  
/// retrieve.  
/// </summary>  
/// <param name="client">The client object used to retrieve the secret  
/// value for the given secret name.</param>  
/// <param name="secretName">The name of the secret value to retrieve.</param>  
/// <returns>The GetSecretValueResponse object returned by  
/// GetSecretValueAsync.</returns>  
public static async Task<GetSecretValueResponse> GetSecretAsync(  
    IAmazonSecretsManager client,  
    string secretName)  
{  
    GetSecretValueRequest request = new();  
    request.SecretId = secretName;  
    request.VersionStage = "AWSCURRENT"; // VersionStage defaults to AWSCURRENT  
if unspecified.  
  
    GetSecretValueResponse response = null;  
  
    // For the sake of simplicity, this example handles only the most  
    // general SecretsManager exception.  
    try  
    {  
        response = await client.GetSecretValueAsync(request);  
    }  
    catch (AmazonSecretsManagerException e)  
    {  
        Console.WriteLine($"Error: {e.Message}");  
    }  
  
    return response;  
}  
  
/// <summary>  
/// Decodes the secret returned by the call to GetSecretValueAsync and  
/// returns it to the calling program.  
/// </summary>  
/// <param name="response">A GetSecretValueResponse object containing  
/// the requested secret value returned by GetSecretValueAsync.</param>  
/// <returns>A string representing the decoded secret value.</returns>  
public static string DecodeString(GetSecretValueResponse response)  
{
```

```
// Decrypts secret using the associated AWS Key Management Service
// Customer Master Key (CMK.) Depending on whether the secret is a
// string or binary value, one of these fields will be populated.
MemoryStream memoryStream = new();

if (response.SecretString is not null)
{
    var secret = response.SecretString;
    return secret;
}
else if (response.SecretBinary is not null)
{
    memoryStream = response.SecretBinary;
    StreamReader reader = new StreamReader(memoryStream);
    string decodedBinarySecret =
        System.Text.Encoding.UTF8.GetString(Convert.FromBase64String(reader.ReadToEnd()));
    return decodedBinarySecret;
}
else
{
    return string.Empty;
}
}
```

- For API details, see [GetSecretValue](#) in *AWS SDK for .NET API Reference*.

## AWS Support examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with AWS Support.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Get started

#### Hello AWS Support

The following code examples show how to get started using AWS Support.

.NET

##### AWS SDK for .NET

###### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon AWSSupport;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

public static class HelloSupport
{
    static async Task Main(string[] args)
```

```
{  
    // Use the AWS .NET Core Setup package to set up dependency injection for  
    // the AWS Support service.  
    // Use your AWS profile name, or leave it blank to use the default profile.  
    // You must have one of the following AWS Support plans: Business,  
    Enterprise On-Ramp, or Enterprise. Otherwise, an exception will be thrown.  
    using var host = Host.CreateDefaultBuilder(args)  
        .ConfigureServices(_=>  
            services.AddAWSService<IAmazonAWSSupport>()  
        ).Build();  
  
    // Now the client is available for injection.  
    var supportClient = host.Services.GetRequiredService<IAmazonAWSSupport>();  
  
    // You can use await and any of the async methods to get a response.  
    var response = await supportClient.DescribeServicesAsync();  
    Console.WriteLine($"\\tHello AWS Support! There are  
{response.Services.Count} services available.");  
}  
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
 * Before running this Java (v2) code example, set up your development environment,  
 including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * In addition, you must have the AWS Business Support Plan to use the AWS Support  
 Java API. For more information, see:  
 *  
 * https://aws.amazon.com/premiumsupport/plans/  
 *  
 * This Java example performs the following task:  
 *  
 * 1. Gets and displays available services.  
 *  
 *  
 * NOTE: To see multiple operations, see SupportScenario.  
 */  
  
public class HelloSupport {  
  
    public static void main(String[] args) {  
        Region region = Region.US_WEST_2;  
        SupportClient supportClient = SupportClient.builder()  
            .region(region)  
            .build();  
  
        System.out.println("***** Step 1. Get and display available services.");  
        displayServices(supportClient);  
    }  
}
```

```
}

// Return a List that contains a Service name and Category name.
public static void displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
    .language("en")
    .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        List<Service> services = response.services();

        System.out.println("Get the first 10 services");
        int index = 1;
        for (Service service: services) {
            if (index== 11)
                break;

            System.out.println("The Service name is: "+service.name());

            // Display the Categories for this service.
            List<Category> categories = service.categories();
            for (Category cat: categories) {
                System.out.println("The category name is: "+cat.name());
            }
            index++ ;
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

In addition, you must have the AWS Business Support Plan to use the AWS Support
Java API. For more information, see:

https://aws.amazon.com/premiumsupport/plans/
```

```
This Kotlin example performs the following task:  
  
1. Gets and displays available services.  
*/  
  
suspend fun main() {  
    displaySomeServices()  
}  
  
// Return a List that contains a Service name and Category name.  
suspend fun displaySomeServices() {  
    val servicesRequest = DescribeServicesRequest {  
        language = "en"  
    }  
  
    SupportClient { region = "us-west-2" }.use { supportClient ->  
        val response = supportClient.describeServices(servicesRequest)  
        println("Get the first 10 services")  
        var index = 1  
  
        response.services?.forEach { service ->  
            if (index == 11) {  
                return@forEach  
            }  
  
            println("The Service name is: " + service.name)  
  
            // Get the categories for this service.  
            service.categories?.forEach { cat ->  
                println("The category name is ${cat.name}")  
                index++  
            }  
        }  
    }  
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for Kotlin API reference*.

## Topics

- [Actions \(p. 2575\)](#)
- [Scenarios \(p. 2581\)](#)

## Actions

### Add a communication to a case

The following code example shows how to add an AWS Support communication with an attachment to a support case.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
```

```
    /// Add communication to a case, including optional attachment set ID and CC email
    /// addresses.
    /// </summary>
    /// <param name="caseId">Id for the support case.</param>
    /// <param name="body">Body text of the communication.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set.</param>
    /// <param name="ccEmailAddresses">Optional list of CC email addresses.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> AddCommunicationToCase(string caseId, string body,
        string? attachmentSetId = null, List<string>? ccEmailAddresses = null)
    {
        var response = await _amazonSupport.AddCommunicationToCaseAsync(
            new AddCommunicationToCaseRequest()
            {
                CaseId = caseId,
                CommunicationBody = body,
                AttachmentSetId = attachmentSetId,
                CcEmailAddresses = ccEmailAddresses
            });
        return response.Result;
    }
```

- For API details, see [AddCommunicationToCase](#) in *AWS SDK for .NET API Reference*.

## Add an attachment to a set

The following code example shows how to add an AWS Support attachment to an attachment set.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    /// <summary>
    /// Add an attachment to a set, or create a new attachment set if one does not
    /// exist.
    /// </summary>
    /// <param name="data">The data for the attachment.</param>
    /// <param name="fileName">The file name for the attachment.</param>
    /// <param name="attachmentSetId">Optional setId for the attachment. Creates a new
    /// attachment set if empty.</param>
    /// <returns>The setId of the attachment.</returns>
    public async Task<string> AddAttachmentToSet(MemoryStream data, string fileName,
        string? attachmentSetId = null)
    {
        var response = await _amazonSupport.AddAttachmentsToSetAsync(
            new AddAttachmentsToSetRequest()
            {
                AttachmentSetId = attachmentSetId,
                Attachments = new List<Attachment>()
                {
                    new Attachment
                    {
                        Data = data,
                        FileName = fileName
                    }
                }
            });
    }
```

```
        return response.AttachmentSetId;
    }
```

- For API details, see [AddAttachmentsToSet](#) in *AWS SDK for .NET API Reference*.

## Create a case

The following code example shows how to create a new AWS Support case.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>
/// Create a new support case.
///</summary>
///<param name="serviceCode">Service code for the new case.</param>
///<param name="categoryCode">Category for the new case.</param>
///<param name="severityCode">Severity code for the new case.</param>
///<param name="subject">Subject of the new case.</param>
///<param name="body">Body text of the new case.</param>
///<param name="language">Optional language support for your case.
/// Currently "en" (English) and "ja" (Japanese) are supported.</param>
///<param name="attachmentSetId">Optional Id for an attachment set for the new
case.</param>
///<param name="issueType">Optional issue type for the new case. Options are
"customer-service" or "technical".</param>
///<returns>The caseId of the new support case.</returns>
public async Task<string> CreateCase(string serviceCode, string categoryCode,
string severityCode, string subject,
string body, string language = "en", string? attachmentSetId = null, string
issueType = "customer-service")
{
    var response = await _amazonSupport.CreateCaseAsync(
        new CreateCaseRequest()
    {
        ServiceCode = serviceCode,
        CategoryCode = categoryCode,
        SeverityCode = severityCode,
        Subject = subject,
        Language = language,
        AttachmentSetId = attachmentSetId,
        IssueType = issueType,
        CommunicationBody = body
    });
    return response.CaseId;
}
```

- For API details, see [CreateCase](#) in *AWS SDK for .NET API Reference*.

## Describe an attachment

The following code example shows how to describe an attachment for an AWS Support case.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get description of a specific attachment.
/// </summary>
/// <param name="attachmentId">Id of the attachment, usually fetched by describing
the communications of a case.</param>
/// <returns>The attachment object.</returns>
public async Task<Attachment> DescribeAttachment(string attachmentId)
{
    var response = await _amazonSupport.DescribeAttachmentAsync(
        new DescribeAttachmentRequest()
    {
        AttachmentId = attachmentId
    });
    return response.Attachment;
}
```

- For API details, see [DescribeAttachment](#) in *AWS SDK for .NET API Reference*.

## Describe cases

The following code example shows how to describe AWS Support cases.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get case details for a list of case ids, optionally with date filters.
/// </summary>
/// <param name="caseIds">The list of case IDs.</param>
/// <param name="displayId">Optional display ID.</param>
/// <param name="includeCommunication">True to include communication. Defaults to
true.</param>
/// <param name="includeResolvedCases">True to include resolved cases. Defaults to
false.</param>
/// <param name="afterTime">The optional start date for a filtered search.</param>
/// <param name="beforeTime">The optional end date for a filtered search.</param>
/// <param name="language">Optional language support for your case.
/// Currently "en" (English) and "ja" (Japanese) are supported.</param>
/// <returns>A list of CaseDetails.</returns>
public async Task<List<CaseDetails>> DescribeCases(List<string> caseIds, string?
displayId = null, bool includeCommunication = true,
bool includeResolvedCases = false, DateTime? afterTime = null, DateTime?
beforeTime = null,
string language = "en")
{
    var results = new List<CaseDetails>();
    var paginateCases = _amazonSupport.Paginator.DescribeCases(
```

```
new DescribeCasesRequest()
{
    CaseIdList = caseIds,
    DisplayId = displayId,
    IncludeCommunications = includeCommunication,
    IncludeResolvedCases = includeResolvedCases,
    AfterTime = afterTime?.ToString("o"),
    BeforeTime = beforeTime?.ToString("o"),
    Language = language
});
// Get the entire list using the paginator.
await foreach (var cases in paginateCases.Cases)
{
    results.Add(cases);
}
return results;
}
```

- For API details, see [DescribeCases](#) in *AWS SDK for .NET API Reference*.

## Describe communications

The following code example shows how to describe AWS Support communications for a case.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Describe the communications for a case, optionally with a date filter.
/// </summary>
/// <param name="caseId">The ID of the support case.</param>
/// <param name="afterTime">The optional start date for a filtered search.</param>
/// <param name="beforeTime">The optional end date for a filtered search.</param>
/// <returns>The list of communications for the case.</returns>
public async Task<List<Communication>> DescribeCommunications(string caseId,
DateTime? afterTime = null, DateTime? beforeTime = null)
{
    var results = new List<Communication>();
    var paginateCommunications = _amazonSupport.Paginator.DescribeCommunications(
        new DescribeCommunicationsRequest()
    {
        CaseId = caseId,
        AfterTime = afterTime?.ToString("G"),
        BeforeTime = beforeTime?.ToString("G")
    });
    // Get the entire list using the paginator.
    await foreach (var communications in paginateCommunications.Communications)
    {
        results.Add(communications);
    }
    return results;
}
```

- For API details, see [DescribeCommunications](#) in *AWS SDK for .NET API Reference*.

## Describe services

The following code example shows how to describe the list of AWS services.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get the descriptions of AWS services.
/// </summary>
/// <param name="name">Optional language for services.
/// Currently "en" (English) and "ja" (Japanese) are supported.</param>
/// <returns>The list of AWS service descriptions.</returns>
public async Task<List<Service>> DescribeServices(string language = "en")
{
    var response = await _amazonSupport.DescribeServicesAsync(
        new DescribeServicesRequest()
    {
        Language = language
    });
    return response.Services;
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for .NET API Reference*.

## Describe severity levels

The following code example shows how to describe AWS Support severity levels.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get the descriptions of support severity levels.
/// </summary>
/// <param name="name">Optional language for severity levels.
/// Currently "en" (English) and "ja" (Japanese) are supported.</param>
/// <returns>The list of support severity levels.</returns>
public async Task<List<SeverityLevel>> DescribeSeverityLevels(string language =
"en")
{
    var response = await _amazonSupport.DescribeSeverityLevelsAsync(
        new DescribeSeverityLevelsRequest()
    {
        Language = language
    });
    return response.SeverityLevels;
}
```

- For API details, see [DescribeSeverityLevels](#) in *AWS SDK for .NET API Reference*.

## Resolve case

The following code example shows how to resolve an AWS Support case.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Resolve a support case by caseId.
/// </summary>
/// <param name="caseId">Id for the support case.</param>
/// <returns>The final status of the case after resolving.</returns>
public async Task<string> ResolveCase(string caseId)
{
    var response = await _amazonSupport.ResolveCaseAsync(
        new ResolveCaseRequest()
    {
        CaseId = caseId
    });
    return response.FinalCaseStatus;
}
```

- For API details, see [ResolveCase](#) in *AWS SDK for .NET API Reference*.

## Scenarios

### Get started with cases

The following code example shows how to:

- Get and display available services and severity levels for cases.
- Create a support case using a selected service, category, and severity level.
- Get and display a list of open cases for the current day.
- Add an attachment set and a communication to the new case.
- Describe the new attachment and communication for the case.
- Resolve the case.
- Get and display a list of resolved cases for the current day.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
/// <summary>
```

```
///> Hello AWS Support example.  
///> </summary>  
public static class SupportCaseScenario  
{  
    /*  
     * Before running this .NET code example, set up your development environment,  
     * including your credentials.  
     * To use the AWS Support API, you must have one of the following AWS Support plans:  
     * Business, Enterprise On-Ramp, or Enterprise.  
  
     * This .NET example performs the following tasks:  
     * 1. Get and display services. Select a service from the list.  
     * 2. Select a category from the selected service.  
     * 3. Get and display severity levels and select a severity level from the list.  
     * 4. Create a support case using the selected service, category, and severity level.  
     * 5. Get and display a list of open support cases for the current day.  
     * 6. Create an attachment set with a sample text file to add to the case.  
     * 7. Add a communication with the attachment to the support case.  
     * 8. List the communications of the support case.  
     * 9. Describe the attachment set.  
     * 10. Resolve the support case.  
     * 11. Get a list of resolved cases for the current day.  
    */  
  
    private static SupportWrapper _supportWrapper = null!;  
  
    static async Task Main(string[] args)  
    {  
        // Set up dependency injection for the AWS Support service.  
        // Use your AWS profile name, or leave it blank to use the default profile.  
        using var host = Host.CreateDefaultBuilder(args)  
            .ConfigureLogging(logging =>  
                logging.AddFilter("System", LogLevel.Debug)  
                    .AddFilter<DebugLoggerProvider>("Microsoft", LogLevel.Information)  
                    .AddFilter<ConsoleLoggerProvider>("Microsoft", LogLevel.Trace))  
            .ConfigureServices(_>, services =>  
                services.AddAWSService<IAmazonAWSSupport>(new AWSOptions() { Profile =  
"default" })  
                    .AddTransient<SupportWrapper>()  
            )  
            .Build();  
  
        var logger = LoggerFactory.Create(builder =>  
        {  
            builder.AddConsole();  
        }).CreateLogger(typeof(SupportCaseScenario));  
  
        _supportWrapper = host.Services.GetRequiredService<SupportWrapper>();  
  
        Console.WriteLine(new string('-', 80));  
        Console.WriteLine("Welcome to the AWS Support case example scenario.");  
        Console.WriteLine(new string('-', 80));  
  
        try  
        {  
            var apiSupported = await _supportWrapper.VerifySubscription();  
            if (!apiSupported)  
            {  
                logger.LogError("You must have a Business, Enterprise On-Ramp, or  
Enterprise Support " +  
                               "plan to use the AWS Support API. \n\nPlease upgrade  
your subscription to run these examples.");  
                return;  
            }  
  
            var service = await DisplayAndSelectServices();  
        }
```

```
        var category = DisplayAndSelectCategories(service);

        var severityLevel = await DisplayAndSelectSeverity();

        var caseId = await CreateSupportCase(service, category, severityLevel);

        await DescribeTodayOpenCases();

        var attachmentSetId = await CreateAttachmentSet();

        await AddCommunicationToCase(attachmentSetId, caseId);

        var attachmentId = await ListCommunicationsForCase(caseId);

        await DescribeCaseAttachment(attachmentId);

        await ResolveCase(caseId);

        await DescribeTodayResolvedCases();

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("AWS Support case example scenario complete.");
        Console.WriteLine(new string('-', 80));
    }
    catch (Exception ex)
    {
        logger.LogError(ex, "There was a problem executing the scenario.");
    }
}

/// <summary>
/// List some available services from AWS Support, and select a service for the
example.
/// </summary>
/// <returns>The selected service.</returns>
private static async Task<Service> DisplayAndSelectServices()
{
    Console.WriteLine(new string('-', 80));
    var services = await _supportWrapper.DescribeServices();
    Console.WriteLine($"AWS Support client returned {services.Count} services.");

    Console.WriteLine($"1. Displaying first 10 services:");
    for (int i = 0; i < 10 && i < services.Count; i++)
    {
        Console.WriteLine($"{i + 1}. {services[i].Name}");
    }

    var choiceNumber = 0;
    while (choiceNumber < 1 || choiceNumber > services.Count)
    {
        Console.WriteLine(
            "Select an example support service by entering a number from the
preceding list:");
        var choice = Console.ReadLine();
        Int32.TryParse(choice, out choiceNumber);
    }
    Console.WriteLine(new string('-', 80));

    return services[choiceNumber - 1];
}

/// <summary>
/// List the available categories for a service and select a category for the
example.
/// </summary>
```

```
    ///<param name="service">Service to use for displaying categories.</param>
    ///<returns>The selected category.</returns>
    private static Category DisplayAndSelectCategories(Service service)
    {
        Console.WriteLine(new string('-', 80));

        Console.WriteLine($"2. Available support categories for Service
\"{service.Name}\":");
        for (int i = 0; i < service.Categories.Count; i++)
        {
            Console.WriteLine($"\\t{i + 1}. {service.Categories[i].Name}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > service.Categories.Count)
        {
            Console.WriteLine(
                "Select an example support category by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        Console.WriteLine(new string('-', 80));

        return service.Categories[choiceNumber - 1];
    }

    ///<summary>
    /// List available severity levels from AWS Support, and select a level for the
example.
    ///</summary>
    ///<returns>The selected severity level.</returns>
    private static async Task<SeverityLevel> DisplayAndSelectSeverity()
    {
        Console.WriteLine(new string('-', 80));
        var severityLevels = await _supportWrapper.DescribeSeverityLevels();

        Console.WriteLine($"3. Get and display available severity levels:");
        for (int i = 0; i < 10 && i < severityLevels.Count; i++)
        {
            Console.WriteLine($"\\t{i + 1}. {severityLevels[i].Name}");
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > severityLevels.Count)
        {
            Console.WriteLine(
                "Select an example severity level by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
        Console.WriteLine(new string('-', 80));

        return severityLevels[choiceNumber - 1];
    }

    ///<summary>
    /// Create an example support case.
    ///</summary>
    ///<param name="service">Service to use for the new case.</param>
    ///<param name="category">Category to use for the new case.</param>
    ///<param name="severity">Severity to use for the new case.</param>
    ///<returns>The caseId of the new support case.</returns>
    private static async Task<string> CreateSupportCase(Service service,
```

```
        Category category, SeverityLevel severity)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. Create an example support case" +
            $" with the following settings:" +
            $" \n\tService: {service.Name}, Category: {category.Name} " +
            $"and Severity Level: {severity.Name}.");
        var caseId = await _supportWrapper.CreateCase(service.Code, category.Code,
severity.Code,
            "Example case for testing, ignore.", "This is my example support case.");

        Console.WriteLine($"\\tNew case created with ID {caseId}");

        Console.WriteLine(new string('-', 80));
        return caseId;
    }

    ///<summary>
    ///<List open cases for the current day.
    ///</summary>
    ///<returns>Async task.</returns>
    private static async Task DescribeTodayOpenCases()
    {
        Console.WriteLine($"5. List the open support cases for the current day.");
        // Describe the cases. If it is empty, try again and allow time for the new
        case to appear.
        List<CaseDetails> currentOpenCases = null!;
        while (currentOpenCases == null || currentOpenCases.Count == 0)
        {
            Thread.Sleep(1000);
            currentOpenCases = await _supportWrapper.DescribeCases(
                new List<string>(),
                null,
                false,
                false,
                DateTime.Today,
                DateTime.Now);
        }

        foreach (var openCase in currentOpenCases)
        {
            Console.WriteLine($"\\tCase: {openCase.CaseId} created
{openCase.TimeCreated}");
        }

        Console.WriteLine(new string('-', 80));
    }

    ///<summary>
    ///<Create an attachment set for a support case.
    ///</summary>
    ///<returns>The attachment set id.</returns>
    private static async Task<string> CreateAttachmentSet()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. Create an attachment set for a support case.");
        var fileName = "example_attachment.txt";

        // Create the file if it does not already exist.
        if (!File.Exists(fileName))
        {
            await using StreamWriter sw = File.CreateText(fileName);
            await sw.WriteLineAsync(
                "This is a sample file for attachment to a support case.");
        }
    }
}
```

```
        await using var ms = new MemoryStream(await File.ReadAllBytesAsync(fileName));

        var attachmentSetId = await _supportWrapper.AddAttachmentToSet(
            ms,
            fileName);

        Console.WriteLine($"\\tNew attachment set created with id: \\n\\t{attachmentSetId.Substring(0, 65)}...");

        Console.WriteLine(new string('-', 80));

        return attachmentSetId;
    }

    ///<summary>
    /// Add an attachment set and communication to a case.
    ///</summary>
    ///<param name="attachmentSetId">Id of the attachment set.</param>
    ///<param name="caseId">Id of the case to receive the attachment set.</param>
    ///<returns>Async task.</returns>
    private static async Task AddCommunicationToCase(string attachmentSetId, string
caseId)
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"7. Add attachment set and communication to {caseId}.");

    await _supportWrapper.AddCommunicationToCase(
        caseId,
        "This is an example communication added to a support case.",
        attachmentSetId);

    Console.WriteLine($"\\tNew attachment set and communication added to {caseId}");

    Console.WriteLine(new string('-', 80));
}

    ///<summary>
    /// List the communications for a case.
    ///</summary>
    ///<param name="caseId">Id of the case to describe.</param>
    ///<returns>An attachment id.</returns>
    private static async Task<string> ListCommunicationsForCase(string caseId)
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"8. List communications for case {caseId}.");

    var communications = await _supportWrapper.DescribeCommunications(caseId);
    var attachmentId = "";
    foreach (var communication in communications)
    {
        Console.WriteLine(
            $"\\tCommunication created on: {communication.TimeCreated} has
{communication.AttachmentSet.Count} attachments.");
        if (communication.AttachmentSet.Any())
        {
            attachmentId = communication.AttachmentSet.First().AttachmentId;
        }
    }

    Console.WriteLine(new string('-', 80));
    return attachmentId;
}

    ///<summary>
    /// Describe an attachment by id.

```

```

    ///</summary>
    ///<param name="attachmentId">Id of the attachment to describe.</param>
    ///<returns>Async task.</returns>
    private static async Task DescribeCaseAttachment(string attachmentId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Describe the attachment set.");

        var attachment = await _supportWrapper.DescribeAttachment(attachmentId);
        var data = Encoding.ASCII.GetString(attachment.Data.ToArray());
        Console.WriteLine($"\\tAttachment includes {attachment.FileName} with data: \\n\\t{data}");

        Console.WriteLine(new string('-', 80));
    }

    ///<summary>
    /// Resolve the support case.
    ///</summary>
    ///<param name="caseId">Id of the case to resolve.</param>
    ///<returns>Async task.</returns>
    private static async Task ResolveCase(string caseId)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Resolve case {caseId}.");

        var status = await _supportWrapper.ResolveCase(caseId);
        Console.WriteLine($"\\tCase {caseId} has final status {status}");

        Console.WriteLine(new string('-', 80));
    }

    ///<summary>
    /// List resolved cases for the current day.
    ///</summary>
    ///<returns>Async Task.</returns>
    private static async Task DescribeTodayResolvedCases()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"11. List the resolved support cases for the current day.");
        var currentCases = await _supportWrapper.DescribeCases(
            new List<string>(),
            null,
            false,
            true,
            DateTime.Today,
            DateTime.Now);

        foreach (var currentCase in currentCases)
        {
            if (currentCase.Status == "resolved")
            {
                Console.WriteLine(
                    $"\\tCase: {currentCase.CaseId}: status {currentCase.Status}");
            }
        }

        Console.WriteLine(new string('-', 80));
    }
}

```

Wrapper methods used by the scenario for AWS Support actions.

```
/// <summary>
/// Wrapper methods to use AWS Support for working with support cases.
/// </summary>
public class SupportWrapper
{
    private readonly IAmazonAWSSupport _amazonSupport;
    public SupportWrapper(IAmazonAWSSupport amazonSupport)
    {
        _amazonSupport = amazonSupport;
    }

    /// <summary>
    /// Get the descriptions of AWS services.
    /// </summary>
    /// <param name="name">Optional language for services.
    /// Currently "en" (English) and "ja" (Japanese) are supported.</param>
    /// <returns>The list of AWS service descriptions.</returns>
    public async Task<List<Service>> DescribeServices(string language = "en")
    {
        var response = await _amazonSupport.DescribeServicesAsync(
            new DescribeServicesRequest()
        {
            Language = language
        });
        return response.Services;
    }

    /// <summary>
    /// Get the descriptions of support severity levels.
    /// </summary>
    /// <param name="name">Optional language for severity levels.
    /// Currently "en" (English) and "ja" (Japanese) are supported.</param>
    /// <returns>The list of support severity levels.</returns>
    public async Task<List<SeverityLevel>> DescribeSeverityLevels(string language =
"en")
    {
        var response = await _amazonSupport.DescribeSeverityLevelsAsync(
            new DescribeSeverityLevelsRequest()
        {
            Language = language
        });
        return response.SeverityLevels;
    }

    /// <summary>
    /// Create a new support case.
    /// </summary>
    /// <param name="serviceCode">Service code for the new case.</param>
    /// <param name="categoryCode">Category for the new case.</param>
    /// <param name="severityCode">Severity code for the new case.</param>
    /// <param name="subject">Subject of the new case.</param>
    /// <param name="body">Body text of the new case.</param>
    /// <param name="language">Optional language support for your case.
    /// Currently "en" (English) and "ja" (Japanese) are supported.</param>
    /// <param name="attachmentSetId">Optional Id for an attachment set for the new
case.</param>
    /// <param name="issueType">Optional issue type for the new case. Options are
"customer-service" or "technical".</param>
    /// <returns>The caseId of the new support case.</returns>
    public async Task<string> CreateCase(string serviceCode, string categoryCode,
string severityCode, string subject,
```

```
        string body, string language = "en", string? attachmentSetId = null, string
issueType = "customer-service")
{
    var response = await _amazonSupport.CreateCaseAsync(
        new CreateCaseRequest()
    {
        ServiceCode = serviceCode,
        CategoryCode = categoryCode,
        SeverityCode = severityCode,
        Subject = subject,
        Language = language,
        AttachmentSetId = attachmentSetId,
        IssueType = issueType,
        CommunicationBody = body
    });
    return response.CaseId;
}

/// <summary>
/// Add an attachment to a set, or create a new attachment set if one does not
exist.
/// </summary>
/// <param name="data">The data for the attachment.</param>
/// <param name="fileName">The file name for the attachment.</param>
/// <param name="attachmentSetId">Optional setId for the attachment. Creates a new
attachment set if empty.</param>
/// <returns>The setId of the attachment.</returns>
public async Task<string> AddAttachmentToSet(MemoryStream data, string fileName,
string? attachmentSetId = null)
{
    var response = await _amazonSupport.AddAttachmentsToSetAsync(
        new AddAttachmentsToSetRequest()
    {
        AttachmentSetId = attachmentSetId,
        Attachments = new List<Attachment>
        {
            new Attachment
            {
                Data = data,
                FileName = fileName
            }
        }
    });
    return response.AttachmentSetId;
}

/// <summary>
/// Get description of a specific attachment.
/// </summary>
/// <param name="attachmentId">Id of the attachment, usually fetched by describing
the communications of a case.</param>
/// <returns>The attachment object.</returns>
public async Task<Attachment> DescribeAttachment(string attachmentId)
{
    var response = await _amazonSupport.DescribeAttachmentAsync(
        new DescribeAttachmentRequest()
    {
        AttachmentId = attachmentId
    });
    return response.Attachment;
}
```

```

    ///<summary>
    /// Add communication to a case, including optional attachment set ID and CC email
    addresses.
    ///</summary>
    ///<param name="caseId">Id for the support case.</param>
    ///<param name="body">Body text of the communication.</param>
    ///<param name="attachmentSetId">Optional Id for an attachment set.</param>
    ///<param name="ccEmailAddresses">Optional list of CC email addresses.</param>
    ///<returns>True if successful.</returns>
    public async Task<bool> AddCommunicationToCase(string caseId, string body,
        string? attachmentSetId = null, List<string>? ccEmailAddresses = null)
    {
        var response = await _amazonSupport.AddCommunicationToCaseAsync(
            new AddCommunicationToCaseRequest()
            {
                CaseId = caseId,
                CommunicationBody = body,
                AttachmentSetId = attachmentSetId,
                CcEmailAddresses = ccEmailAddresses
            });
        return response.Result;
    }

    ///<summary>
    /// Describe the communications for a case, optionally with a date filter.
    ///</summary>
    ///<param name="caseId">The ID of the support case.</param>
    ///<param name="afterTime">The optional start date for a filtered search.</param>
    ///<param name="beforeTime">The optional end date for a filtered search.</param>
    ///<returns>The list of communications for the case.</returns>
    public async Task<List<Communication>> DescribeCommunications(string caseId,
        DateTime? afterTime = null, DateTime? beforeTime = null)
    {
        var results = new List<Communication>();
        var paginateCommunications = _amazonSupport.Paginator.DescribeCommunications(
            new DescribeCommunicationsRequest()
            {
                CaseId = caseId,
                AfterTime = afterTime?.ToString("G"),
                BeforeTime = beforeTime?.ToString("G")
            });
        // Get the entire list using the paginator.
        await foreach (var communications in paginateCommunications.Communications)
        {
            results.Add(communications);
        }
        return results;
    }

    ///<summary>
    /// Get case details for a list of case ids, optionally with date filters.
    ///</summary>
    ///<param name="caseIds">The list of case IDs.</param>
    ///<param name="displayId">Optional display ID.</param>
    ///<param name="includeCommunication">True to include communication. Defaults to
    true.</param>
    ///<param name="includeResolvedCases">True to include resolved cases. Defaults to
    false.</param>
    ///<param name="afterTime">The optional start date for a filtered search.</param>
    ///<param name="beforeTime">The optional end date for a filtered search.</param>

```

```
    /// <param name="language">Optional language support for your case.  
    /// Currently "en" (English) and "ja" (Japanese) are supported.</param>  
    /// <returns>A list of CaseDetails.</returns>  
    public async Task<List<CaseDetails>> DescribeCases(List<string> caseIds, string?  
displayId = null, bool includeCommunication = true,  
        bool includeResolvedCases = false, DateTime? afterTime = null, DateTime?  
beforeTime = null,  
        string language = "en")  
{  
    var results = new List<CaseDetails>();  
    var paginateCases = _amazonSupport.Paginator.DescribeCases(  
        new DescribeCasesRequest()  
    {  
        CaseIdList = caseIds,  
        DisplayId = displayId,  
        IncludeCommunications = includeCommunication,  
        IncludeResolvedCases = includeResolvedCases,  
        AfterTime = afterTime?.ToString("o"),  
        BeforeTime = beforeTime?.ToString("o"),  
        Language = language  
    });  
    // Get the entire list using the paginator.  
    await foreach (var cases in paginateCases.Cases)  
    {  
        results.Add(cases);  
    }  
    return results;  
}  
  
    /// <summary>  
    /// Resolve a support case by caseId.  
    /// </summary>  
    /// <param name="caseId">Id for the support case.</param>  
    /// <returns>The final status of the case after resolving.</returns>  
    public async Task<string> ResolveCase(string caseId)  
{  
    var response = await _amazonSupport.ResolveCaseAsync(  
        new ResolveCaseRequest()  
    {  
        CaseId = caseId  
    });  
    return response.FinalCaseStatus;  
}  
  
    /// <summary>  
    /// Verify the support level for AWS Support API access.  
    /// </summary>  
    /// <returns>True if the subscription level supports API access.</returns>  
    public async Task<bool> VerifySubscription()  
{  
    try  
    {  
        var response = await _amazonSupport.DescribeServicesAsync(  
            new DescribeServicesRequest()  
        {  
            Language = "en"  
        });  
        return response.HttpStatusCode == HttpStatusCode.OK;  
    }  
    catch (Amazon.AWSSupport.AmazonAWSSupportException ex)  
    {  
        if (ex.ErrorCode == "SubscriptionRequiredException")  
        {  
            throw;  
        }  
    }  
}
```

```
        return false;
    }
    else throw;
}
}
```

- For API details, see the following topics in *AWS SDK for .NET API Reference*.
  - [AddAttachmentsToSet](#)
  - [AddCommunicationToCase](#)
  - [CreateCase](#)
  - [DescribeAttachment](#)
  - [DescribeCases](#)
  - [DescribeCommunications](#)
  - [DescribeServices](#)
  - [DescribeSeverityLevels](#)
  - [ResolveCase](#)

## Amazon Transcribe examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon Transcribe.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2592\)](#)

## Actions

### Create a custom vocabulary

The following code example shows how to create a custom Amazon Transcribe vocabulary.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Create a custom vocabulary using a list of phrases. Custom vocabularies
/// improve transcription accuracy for one or more specific words.
/// </summary>
/// <param name="languageCode">The language code of the vocabulary.</param>
/// <param name="phrases">Phrases to use in the vocabulary.</param>
/// <param name="vocabularyName">Name for the vocabulary.</param>
/// <returns>The state of the custom vocabulary.</returns>
public async Task<VocabularyState> CreateCustomVocabulary(LanguageCode
languageCode,
```

```
List<string> phrases, string vocabularyName)
{
    var response = await _amazonTranscribeService.CreateVocabularyAsync(
        new CreateVocabularyRequest
    {
        LanguageCode = languageCode,
        Phrases = phrases,
        VocabularyName = vocabularyName
    });
    return response.VocabularyState;
}
```

- For API details, see [CreateVocabulary](#) in *AWS SDK for .NET API Reference*.

## Delete a custom vocabulary

The following code example shows how to delete a custom Amazon Transcribe vocabulary.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Delete an existing custom vocabulary.
/// </summary>
/// <param name="vocabularyName">Name of the vocabulary to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteCustomVocabulary(string vocabularyName)
{
    var response = await _amazonTranscribeService.DeleteVocabularyAsync(
        new DeleteVocabularyRequest
    {
        VocabularyName = vocabularyName
    });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- For API details, see [DeleteVocabulary](#) in *AWS SDK for .NET API Reference*.

## Delete a medical transcription job

The following code example shows how to delete an Amazon Transcribe Medical transcription job.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
```

```
    /// Delete a medical transcription job. Also deletes the transcript associated with
    /// the job.
    /// </summary>
    /// <param name="jobName">Name of the medical transcription job to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMedicalTranscriptionJob(string jobName)
    {
        var response = await
_amazonTranscribeService.DeleteMedicalTranscriptionJobAsync(
            new DeleteMedicalTranscriptionJobRequest()
            {
                MedicalTranscriptionJobName = jobName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- For API details, see [DeleteMedicalTranscriptionJob](#) in *AWS SDK for .NET API Reference*.

## Delete a transcription job

The following code example shows how to delete an Amazon Transcribe transcription job.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    /// <summary>
    /// Delete a transcription job. Also deletes the transcript associated with the
    /// job.
    /// </summary>
    /// <param name="jobName">Name of the transcription job to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTranscriptionJob(string jobName)
    {
        var response = await _amazonTranscribeService.DeleteTranscriptionJobAsync(
            new DeleteTranscriptionJobRequest()
            {
                TranscriptionJobName = jobName
            });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- For API details, see [DeleteTranscriptionJob](#) in *AWS SDK for .NET API Reference*.

## Get a custom vocabulary

The following code example shows how to get a custom Amazon Transcribe vocabulary.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get information about a custom vocabulary.
/// </summary>
/// <param name="vocabularyName">Name of the vocabulary.</param>
/// <returns>The state of the custom vocabulary.</returns>
public async Task<VocabularyState> GetCustomVocabulary(string vocabularyName)
{
    var response = await _amazonTranscribeService.GetVocabularyAsync(
        new GetVocabularyRequest()
    {
        VocabularyName = vocabularyName
    });
    return response.VocabularyState;
}
```

- For API details, see [GetVocabulary](#) in *AWS SDK for .NET API Reference*.

## Get a transcription job

The following code example shows how to get an Amazon Transcribe transcription job.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Get details about a transcription job.
/// </summary>
/// <param name="jobName">A unique name for the transcription job.</param>
/// <returns>A TranscriptionJob instance with information on the requested job.</returns>
public async Task<TranscriptionJob> GetTranscriptionJob(string jobName)
{
    var response = await _amazonTranscribeService.GetTranscriptionJobAsync(
        new GetTranscriptionJobRequest()
    {
        TranscriptionJobName = jobName
    });
    return response.TranscriptionJob;
}
```

- For API details, see [GetTranscriptionJob](#) in *AWS SDK for .NET API Reference*.

## List custom vocabularies

The following code example shows how to list custom Amazon Transcribe vocabularies.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    /// <summary>
    /// List custom vocabularies for the current account. Optionally specify a name
    /// filter and a specific state to filter the vocabularies list.
    /// </summary>
    /// <param name="nameContains">Optional string the vocabulary name must contain.</param>
    /// <param name="stateEquals">Optional state of the vocabulary.</param>
    /// <returns>List of information about the vocabularies.</returns>
    public async Task<List<VocabularyInfo>> ListCustomVocabularies(string? nameContains
= null,
    VocabularyState? stateEquals = null)
    {
        var response = await _amazonTranscribeService.ListVocabulariesAsync(
            new ListVocabulariesRequest()
            {
                NameContains = nameContains,
                StateEquals = stateEquals
            });
        return response.Vocabularies;
    }
```

- For API details, see [ListVocabularies](#) in *AWS SDK for .NET API Reference*.

## List medical transcription jobs

The following code example shows how to list Amazon Transcribe Medical transcription jobs.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    /// <summary>
    /// List medical transcription jobs, optionally with a name filter.
    /// </summary>
    /// <param name="jobNameContains">Optional name filter for the medical
    transcription jobs.</param>
    /// <returns>A list of summaries about medical transcription jobs.</returns>
    public async Task<List<MedicalTranscriptionJobSummary>> ListMedicalTranscriptionJobs(
        string? jobNameContains = null)
    {
        var response = await
    _amazonTranscribeService.ListMedicalTranscriptionJobsAsync(
            new ListMedicalTranscriptionJobsRequest()
            {
                JobNameContains = jobNameContains
            });
        return response.MedicalTranscriptionJobSummaries;
    }
```

- For API details, see [ListMedicalTranscriptionJobs](#) in *AWS SDK for .NET API Reference*.

## List transcription jobs

The following code example shows how to list Amazon Transcribe transcription jobs.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// List transcription jobs, optionally with a name filter.
/// </summary>
/// <param name="jobNameContains">Optional name filter for the transcription
jobs.</param>
/// <returns>A list of transcription job summaries.</returns>
public async Task<List<TranscriptionJobSummary>> ListTranscriptionJobs(string?
jobNameContains = null)
{
    var response = await _amazonTranscribeService.ListTranscriptionJobsAsync(
        new ListTranscriptionJobsRequest()
    {
        JobNameContains = jobNameContains
    });
    return response.TranscriptionJobSummaries;
}
```

- For API details, see [ListTranscriptionJobs](#) in *AWS SDK for .NET API Reference*.

## Start a medical transcription job

The following code example shows how to start an Amazon Transcribe Medical transcription job.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Start a medical transcription job for a media file. This method returns
/// as soon as the job is started.
/// </summary>
/// <param name="jobName">A unique name for the medical transcription job.</param>
/// <param name="mediaFileUri">The URI of the media file, typically an Amazon S3
location.</param>
/// <param name="mediaFormat">The format of the media file.</param>
/// <param name="outputBucketName">Location for the output, typically an Amazon S3
location.</param>
/// <param name="transcriptionType">Conversation or dictation transcription type.</
param>
/// <returns>A MedicalTransactionJob instance with information on the new job.</
returns>
public async Task<MedicalTransactionJob> StartMedicalTranscriptionJob(
    string jobName, string mediaFileUri,
    MediaFormat mediaFormat, string outputBucketName, Amazon.TranscribeService.Type
transcriptionType)
```

```
{  
    var response = await  
_amazonTranscribeService.StartMedicalTranscriptionJobAsync(  
    new StartMedicalTranscriptionJobRequest()  
    {  
        MedicalTranscriptionJobName = jobName,  
        Media = new Media()  
        {  
            MediaFileUri = mediaFileUri  
        },  
        MediaFormat = mediaFormat,  
        LanguageCode =  
            LanguageCode  
            .EnUS, // The value must be en-US for medical transcriptions.  
        OutputBucketName = outputBucketName,  
        OutputKey =  
            jobName, // The value is a key used to fetch the output of the  
transcription.  
        Specialty = Specialty.PRIMARYCARE, // The value PRIMARYCARE must be  
set.  
        Type = transcriptionType  
    });  
    return response.MedicalTranscriptionJob;  
}
```

- For API details, see [StartMedicalTranscriptionJob](#) in *AWS SDK for .NET API Reference*.

## Start a transcription job

The following code example shows how to start an Amazon Transcribe transcription job.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
///<summary>  
/// Start a transcription job for a media file. This method returns  
/// as soon as the job is started.  
///</summary>  
///<param name="jobName">A unique name for the transcription job.</param>  
///<param name="mediaFileUri">The URI of the media file, typically an Amazon S3  
location.</param>  
///<param name="mediaFormat">The format of the media file.</param>  
///<param name="languageCode">The language code of the media file, such as en-  
US.</param>  
///<param name="vocabularyName">Optional name of a custom vocabulary.</param>  
///<returns>A TranscriptionJob instance with information on the new job.</returns>  
public async Task<TranscriptionJob> StartTranscriptionJob(string jobName, string  
mediaFileUri,  
    MediaFormat mediaFormat, LanguageCode languageCode, string? vocabularyName)  
{  
    var response = await _amazonTranscribeService.StartTranscriptionJobAsync(  
    new StartTranscriptionJobRequest()  
    {  
        TranscriptionJobName = jobName,  
        Media = new Media()  
    }
```

```
        MediaFileUri = mediaFileUri
    },
    MediaFormat = mediaFormat,
    LanguageCode = languageCode,
    Settings = vocabularyName != null ? new Settings()
    {
        VocabularyName = vocabularyName
    } : null
);
return response.TranscriptionJob;
}
```

- For API details, see [StartTranscriptionJob](#) in *AWS SDK for .NET API Reference*.

## Update a custom vocabulary

The following code example shows how to update a custom Amazon Transcribe vocabulary.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// <summary>
/// Update a custom vocabulary with new values. Update overwrites all existing
/// information.
/// </summary>
/// <param name="languageCode">The language code of the vocabulary.</param>
/// <param name="phrases">Phrases to use in the vocabulary.</param>
/// <param name="vocabularyName">Name for the vocabulary.</param>
/// <returns>The state of the custom vocabulary.</returns>
public async Task<VocabularyState> UpdateCustomVocabulary(LanguageCode
languageCode,
    List<string> phrases, string vocabularyName)
{
    var response = await _amazonTranscribeService.UpdateVocabularyAsync(
        new UpdateVocabularyRequest()
    {
        LanguageCode = languageCode,
        Phrases = phrases,
        VocabularyName = vocabularyName
    });
    return response.VocabularyState;
}
```

- For API details, see [UpdateVocabulary](#) in *AWS SDK for .NET API Reference*.

## Amazon Translate examples using AWS SDK for .NET

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for .NET with Amazon Translate.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Topics

- [Actions \(p. 2600\)](#)

## Actions

### Describe a translation job

The following code example shows how to describe an Amazon Translate translation job.

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

public class DescribeTextTranslation
{
    public static async Task Main()
    {
        var client = new AmazonTranslateClient();

        // The Job Id is generated when the text translation job is started
        // with a call to the StartTextTranslationJob method.
        var jobId = "1234567890abcdef01234567890abcde";

        var request = new DescribeTextTranslationJobRequest
        {
            JobId = jobId,
        };

        var jobProperties = await DescribeTranslationJobAsync(client, request);

        DisplayTranslationJobDetails(jobProperties);
    }

    ///<summary>
    /// Retrieve information about an Amazon Translate text translation job.
    ///</summary>
    ///<param name="client">The initialized Amazon Translate client object.</param>
    ///<param name="request">The DescribeTextTranslationJobRequest object.</param>
    ///<returns>The TextTranslationJobProperties object containing
    /// information about the text translation job..</returns>
    public static async Task<TextTranslationJobProperties>
DescribeTranslationJobAsync(
    AmazonTranslateClient client,
    DescribeTextTranslationJobRequest request)
{
    var response = await client.DescribeTextTranslationJobAsync(request);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        return response.TextTranslationJobProperties;
    }
}
```

```
        else
        {
            return null;
        }
    }

    ///<summary>
    /// Displays the properties of the text translation job.
    ///</summary>
    ///<param name="jobProperties">The properties of the text translation
    /// job returned by the call to DescribeTextTranslationJobAsync.</param>
    public static void DisplayTranslationJobDetails(TextTranslationJobProperties
jobProperties)
{
    if (jobProperties is null)
    {
        Console.WriteLine("No text translation job properties found.");
        return;
    }

    // Display the details of the text translation job.
    Console.WriteLine($"{jobProperties.JobId}: {jobProperties.JobName}");
}
}
```

- For API details, see [DescribeTextTranslationJob](#) in *AWS SDK for .NET API Reference*.

## List translation jobs

The following code example shows how to list the Amazon Translate translation jobs.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

public class ListTranslationJobs
{
    public static async Task Main()
    {
        var client = new AmazonTranslateClient();
        var filter = new TextTranslationJobFilter
        {
            JobStatus = "COMPLETED",
        };

        var request = new ListTextTranslationJobsRequest
        {
            MaxResults = 10,
            Filter = filter,
        };

        await ListJobsAsync(client, request);
    }
}
```

```
}

/// <summary>
/// List Amazon Translate text translation jobs.
/// </summary>
/// <param name="client">The initialized Amazon Translate client object.</param>
public static async Task ListJobsAsync(
    AmazonTranslateClient client,
    ListTextTranslationJobsRequest request)
{
    ListTextTranslationJobsResponse response;

    do
    {
        response = await client.ListTextTranslationJobsAsync(request);
        ShowTranslationJobDetails(response.TextTranslationJobPropertiesList);

        request.NextToken = response.NextToken;
    }
    while (response.NextToken is not null);
}

/// <summary>
/// List existing translation job details.
/// </summary>
/// <param name="properties">A list of Amazon Translate text
/// translation jobs.</param>
public static void ShowTranslationJobDetails(List<TextTranslationJobProperties>
properties)
{
    properties.ForEach(prop =>
    {
        Console.WriteLine($"{{prop.JobId}}: {{prop.JobName}}");
        Console.WriteLine($"Status: {{prop.JobStatus}}");
        Console.WriteLine($"Submitted time: {{prop.SubmittedTime}}");
    });
}
```

- For API details, see [ListTextTranslationJobs](#) in *AWS SDK for .NET API Reference*.

## Start a translation job

The following code example shows how to start an Amazon Translate translation job.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;
```

```
public class BatchTranslate
{
    public static async Task Main()
    {
        var contentType = "text/plain";

        // Set this variable to an S3 bucket location with a folder.
        // Input files must be in a folder and not at the bucket root.
        var s3InputUri = "s3://DOC-EXAMPLE-BUCKET1/FOLDER/";
        var s3OutputUri = "s3://DOC-EXAMPLE-BUCKET2/";

        and
        stored. // This role must have permissions to read the source bucket and to read
        // write to the destination bucket where the translated text will be
        var dataAccessRoleArn = "arn:aws:iam::0123456789ab:role/S3TranslateRole";

        var client = new AmazonTranslateClient();

        var inputConfig = new InputDataConfig
        {
            ContentType = contentType,
            S3Uri = s3InputUri,
        };

        var outputConfig = new OutputDataConfig
        {
            S3Uri = s3OutputUri,
        };

        var request = new StartTextTranslationJobRequest
        {
            JobName = "ExampleTranslationJob",
            DataAccessRoleArn = dataAccessRoleArn,
            InputDataConfig = inputConfig,
            OutputDataConfig = outputConfig,
            SourceLanguageCode = "en",
            TargetLanguageCodes = new List<string> { "fr" },
        };

        var response = await StartTextTranslationAsync(client, request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"{response.JobId}: {response.JobStatus}");
        }
    }

    ///<summary>
    /// Start the Amazon Translate text translation job.
    ///</summary>
    ///<param name="client">The initialized AmazonTranslateClient object.</param>
    ///<param name="request">The request object that includes details such
    ///as source and destination bucket names and the IAM Role that will
    ///be used to access the buckets.</param>
    ///<returns>The StartTextTranslationResponse object that includes the
    ///details of the request response.</returns>
    public static async Task<StartTextTranslationJobResponse>
StartTextTranslationAsync(AmazonTranslateClient client, StartTextTranslationJobRequest
request)
    {
        var response = await client.StartTextTranslationJobAsync(request);
        return response;
    }
}
```

- For API details, see [StartTextTranslationJob](#) in *AWS SDK for .NET API Reference*.

## Stop a translation job

The following code example shows how to stop an Amazon Translate translation job.

### AWS SDK for .NET

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.Threading.Tasks;
using Amazon.Translate;
using Amazon.Translate.Model;

class StopTextTranslationJob
{
    public static async Task Main()
    {
        var client = new AmazonTranslateClient();
        var jobId = "1234567890abcdef01234567890abcde";

        var request = new StopTextTranslationJobRequest
        {
            JobId = jobId,
        };

        await StopTranslationJobAsync(client, request);
    }

    ///<summary>
    /// Sends a request to stop a text translation job.
    ///</summary>
    ///<param name="client">Initialized AmazonTrnslateClient object.</param>
    ///<param name="request">The request object to be passed to the
    /// StopTextJobAsync method.</param>
    public static async Task StopTranslationJobAsync(
        AmazonTranslateClient client,
        StopTextTranslationJobRequest request)
    {
        var response = await client.StopTextTranslationJobAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"'{response.JobId}' as status: {response.JobStatus}");
        }
    }
}
```

- For API details, see [StopTextTranslationJob](#) in *AWS SDK for .NET API Reference*.

## Translate text

The following code example shows how to translate text with Amazon Translate.

## AWS SDK for .NET

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;
using Amazon.Translate;
using Amazon.Translate.Model;

public class TranslateText
{
    public static async Task Main()
    {
        // If the region you want to use is different from the region
        // defined for the default user, supply it as a parameter to the
        // Amazon Translate client object constructor.
        var client = new AmazonTranslateClient();

        // Set the source language to "auto" to request Amazon Translate to
        // automatically detect the language of the source text.

        // You can get a list of the languages supported by Amazon Translate
        // in the Amazon Translate Developer's Guide here:
        //     https://docs.aws.amazon.com/translate/latest/dg/what-is.html
        string srcLang = "en"; // English.
        string destLang = "fr"; // French.

        // The Amazon Simple Storage Service (Amazon S3) bucket where the
        // source text file is stored.
        string srcBucket = "DOC-EXAMPLE-BUCKET";
        string srcTextFile = "source.txt";

        var srcText = await GetSourceTextAsync(srcBucket, srcTextFile);
        var destText = await TranslatingTextAsync(client, srcLang, destLang,
srcText);

        ShowText(srcText, destText);
    }

    /// <summary>
    /// Use the Amazon S3 TransferUtility to retrieve the text to translate
    /// from an object in an S3 bucket.
    /// </summary>
    /// <param name="srcBucket">The name of the S3 bucket where the
    /// text is stored.
    /// </param>
    /// <param name="srcTextFile">The key of the S3 object that
    /// contains the text to translate.</param>
    /// <returns>A string representing the source text.</returns>
    public static async Task<string> GetSourceTextAsync(string srcBucket, string
srcTextFile)
    {
        string srcText = string.Empty;

        var s3Client = new AmazonS3Client();
        TransferUtility utility = new TransferUtility(s3Client);

        using var stream = await utility.OpenStreamAsync(srcBucket, srcTextFile);
```

```
StreamReader file = new System.IO.StreamReader(stream);

srcText = file.ReadToEnd();
return srcText;
}

/// <summary>
/// Use the Amazon Translate Service to translate the document from the
/// source language to the specified destination language.
/// </summary>
/// <param name="client">The Amazon Translate Service client used to
/// perform the translation.</param>
/// <param name="srcLang">The language of the source text.</param>
/// <param name="destLang">The destination language for the translated
/// text.</param>
/// <param name="text">A string representing the text to translate.</param>
/// <returns>The text that has been translated to the destination
/// language.</returns>
public static async Task<string> TranslatingTextAsync(AmazonTranslateClient
client, string srcLang, string destLang, string text)
{
    var request = new TranslateTextRequest
    {
        SourceLanguageCode = srcLang,
        TargetLanguageCode = destLang,
        Text = text,
    };

    var response = await client.TranslateTextAsync(request);

    return response.TranslatedText;
}

/// <summary>
/// Show the original text followed by the translated text.
/// </summary>
/// <param name="srcText">The original text to be translated.</param>
/// <param name="destText">The translated text.</param>
public static void ShowText(string srcText, string destText)
{
    Console.WriteLine("Source text:");
    Console.WriteLine(srcText);
    Console.WriteLine();
    Console.WriteLine("Translated text:");
    Console.WriteLine(destText);
}
}
```

- For API details, see [TranslateText](#) in *AWS SDK for .NET API Reference*.

## Cross-service examples using AWS SDK for .NET

The following sample applications use the AWS SDK for .NET to work across multiple AWS services.

### Examples

- [Build a publish and subscription application that translates messages \(p. 2607\)](#)
- [Create a web application to track DynamoDB data \(p. 2607\)](#)
- [Create an Aurora Serverless work item tracker \(p. 2607\)](#)
- [Detect objects in images with Amazon Rekognition using an AWS SDK \(p. 2608\)](#)

## Build a publish and subscription application that translates messages

### AWS SDK for .NET

Shows how to use the Amazon Simple Notification Service .NET API to create a web application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon SNS
- Amazon Translate

## Create a web application to track DynamoDB data

### AWS SDK for .NET

Shows how to use the Amazon DynamoDB .NET API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon SES

## Create an Aurora Serverless work item tracker

### AWS SDK for .NET

Shows how to use the AWS SDK for .NET to create a web application that tracks work items in an Amazon Aurora database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with a RESTful .NET backend.

- Integrate a React web application with AWS services.
- List, add, update, and delete items in an Aurora table.
- Send an email report of filtered work items using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

## Detect objects in images with Amazon Rekognition using an AWS SDK

### AWS SDK for .NET

Shows how to use Amazon Rekognition .NET API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Code examples for SDK for C++

The code examples in this topic show you how to use the AWS SDK for C++ with AWS.

### Examples

- [Single-service actions and scenarios using SDK for C++ \(p. 2608\)](#)
- [Cross-service examples using SDK for C++ \(p. 2714\)](#)

## Single-service actions and scenarios using SDK for C++

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for C++ with AWS services.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Services

- [CloudWatch examples using SDK for C++ \(p. 2609\)](#)
- [CloudWatch Logs examples using SDK for C++ \(p. 2615\)](#)
- [DynamoDB examples using SDK for C++ \(p. 2618\)](#)
- [EventBridge examples using SDK for C++ \(p. 2655\)](#)
- [IAM examples using SDK for C++ \(p. 2657\)](#)
- [Amazon S3 examples using SDK for C++ \(p. 2681\)](#)
- [Amazon SNS examples using SDK for C++ \(p. 2698\)](#)
- [AWS STS examples using SDK for C++ \(p. 2708\)](#)
- [Secrets Manager examples using SDK for C++ \(p. 2709\)](#)
- [Amazon Transcribe examples using SDK for C++ \(p. 2711\)](#)

## CloudWatch examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon CloudWatch.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2609\)](#)

## Actions

### Create an alarm that watches a metric

The following code example shows how to create an Amazon CloudWatch alarm that watches a metric.

#### SDK for C++

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Create the alarm to watch the metric.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::PutMetricAlarmRequest request;
request.SetAlarmName(alarm_name);
request.SetComparisonOperator(
    Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
request.SetEvaluationPeriods(1);
request.SetMetricName("CPUUtilization");
request.SetNamespace("AWS/EC2");
request.SetPeriod(60);
request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
request.SetThreshold(70.0);
request.SetActionsEnabled(false);
request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("InstanceId");
dimension.SetValue(instanceId);

request.AddDimensions(dimension);

auto outcome = cw.PutMetricAlarm(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch alarm:" <<
        outcome.GetError().GetMessage() << std::endl;
```

```
        }
    else
    {
        std::cout << "Successfully created CloudWatch alarm " << alarm_name
                << std::endl;
    }
}
```

- For API details, see [PutMetricAlarm in AWS SDK for C++ API Reference](#).

## Delete alarms

The following code example shows how to delete Amazon CloudWatch alarms.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DeleteAlarmsRequest.h>
#include <iostream>
```

Delete the alarm.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::DeleteAlarmsRequest request;
request.AddAlarmNames(alarm_name);

auto outcome = cw.DeleteAlarms(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to delete CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully deleted CloudWatch alarm " << alarm_name
                << std::endl;
}
```

- For API details, see [DeleteAlarms in AWS SDK for C++ API Reference](#).

## Describe alarms for a metric

The following code example shows how to describe Amazon CloudWatch alarms for a metric.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DescribeAlarmsRequest.h>
#include <aws/monitoring/model/DescribeAlarmsResult.h>
#include <iomanip>
#include <iostream>
```

Describe the alarms.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::DescribeAlarmsRequest request;
request.SetMaxRecords(1);

bool done = false;
bool header = false;
while (!done)
{
    auto outcome = cw.DescribeAlarms(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to describe CloudWatch alarms:" <<
            outcome.GetError().GetMessage() << std::endl;
        break;
    }

    if (!header)
    {
        std::cout << std::left <<
            std::setw(32) << "Name" <<
            std::setw(64) << "Arn" <<
            std::setw(64) << "Description" <<
            std::setw(20) << "LastUpdated" <<
            std::endl;
        header = true;
    }

    const auto &alarms = outcome.GetResult().GetMetricAlarms();
    for (const auto &alarm : alarms)
    {
        std::cout << std::left <<
            std::setw(32) << alarm.GetAlarmName() <<
            std::setw(64) << alarm.GetAlarmArn() <<
            std::setw(64) << alarm.GetAlarmDescription() <<
            std::setw(20) <<
            alarm.GetAlarmConfigurationUpdatedTimestamp().ToGmtString(
                SIMPLE_DATE_FORMAT_STR) <<
            std::endl;
    }

    const auto &next_token = outcome.GetResult().GetNextToken();
    request.SetNextToken(next_token);
    done = next_token.empty();
}
```

- For API details, see [DescribeAlarmsForMetric](#) in *AWS SDK for C++ API Reference*.

### Disable alarm actions

The following code example shows how to disable Amazon CloudWatch alarm actions.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DisableAlarmActionsRequest.h>
#include <iostream>
```

Disable the alarm actions.

```
Aws::CloudWatch::CloudWatchClient cw;

Aws::CloudWatch::Model::DisableAlarmActionsRequest disableAlarmActionsRequest;
disableAlarmActionsRequest.AddAlarmNames(alarm_name);

auto disableAlarmActionsOutcome =
cw.DisableAlarmActions(disableAlarmActionsRequest);
if (!disableAlarmActionsOutcome.IsSuccess())
{
    std::cout << "Failed to disable actions for alarm " << alarm_name <<
        ":" << disableAlarmActionsOutcome.GetError().GetMessage() <<
        std::endl;
}
else
{
    std::cout << "Successfully disabled actions for alarm " <<
        alarm_name << std::endl;
}
```

- For API details, see [DisableAlarmActions](#) in *AWS SDK for C++ API Reference*.

## Enable alarm actions

The following code example shows how to enable Amazon CloudWatch alarm actions.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/EnableAlarmActionsRequest.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

Enable the alarm actions.

```
Aws::CloudWatch::CloudWatchClient cw;
```

```
Aws::CloudWatch::Model::PutMetricAlarmRequest request;
request.SetAlarmName(alarm_name);
request.SetComparisonOperator(
    Aws::CloudWatch::Model::ComparisonOperator::GreaterThanOrEqualToThreshold);
request.SetEvaluationPeriods(1);
request.SetMetricName("CPUUtilization");
request.SetNamespace("AWS/EC2");
request.SetPeriod(60);
request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
request.SetThreshold(70.0);
request.SetActionsEnabled(false);
request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);
request.AddAlarmActions(actionArn);

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("InstanceId");
dimension.SetValue(instanceId);
request.AddDimensions(dimension);

auto outcome = cw.PutMetricAlarm(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch alarm:" <<
        outcome.GetError().GetMessage() << std::endl;
    return;
}

Aws::CloudWatch::Model::EnableAlarmActionsRequest enable_request;
enable_request.AddAlarmNames(alarm_name);

auto enable_outcome = cw.EnableAlarmActions(enable_request);
if (!enable_outcome.IsSuccess())
{
    std::cout << "Failed to enable alarm actions:" <<
        enable_outcome.GetError().GetMessage() << std::endl;
    return;
}

std::cout << "Successfully created alarm " << alarm_name <<
    " and enabled actions on it." << std::endl;
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for C++ API Reference*.

## List metrics

The following code example shows how to list Amazon CloudWatch metrics.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/ListMetricsRequest.h>
#include <aws/monitoring/model/ListMetricsResult.h>
#include <iomanip>
#include <iostream>
```

List the metrics.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::ListMetricsRequest request;

if (argc > 1)
{
    request.SetMetricName(argv[1]);
}

if (argc > 2)
{
    request.SetNamespace(argv[2]);
}

bool done = false;
bool header = false;
while (!done)
{
    auto outcome = cw.ListMetrics(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to list CloudWatch metrics:" <<
                    outcome.GetError().GetMessage() << std::endl;
        break;
    }

    if (!header)
    {
        std::cout << std::left << std::setw(48) << "MetricName" <<
                    std::setw(32) << "Namespace" << "DimensionNameValuePair" <<
                    std::endl;
        header = true;
    }

    const auto &metrics = outcome.GetResult().GetMetrics();
    for (const auto &metric : metrics)
    {
        std::cout << std::left << std::setw(48) <<
                    metric.GetMetricName() << std::setw(32) <<
                    metric.GetNamespace();
        const auto &dimensions = metric.GetDimensions();
        for (auto iter = dimensions.cbegin();  

                iter != dimensions.cend(); ++iter)
        {
            const auto &dimkv = *iter;
            std::cout << dimkv.GetName() << " = " << dimkv.GetValue();
            if (iter + 1 != dimensions.cend())
            {
                std::cout << ", ";
            }
        }
        std::cout << std::endl;
    }

    const auto &next_token = outcome.GetResult().GetNextToken();
    request.SetNextToken(next_token);
    done = next_token.empty();
}
```

- For API details, see [ListMetrics](#) in *AWS SDK for C++ API Reference*.

## Put data into a metric

The following code example shows how to put data into a Amazon CloudWatch metric.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricDataRequest.h>
#include <iostream>
```

Put data into the metric.

```
Aws::CloudWatch::CloudWatchClient cw;

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("UNIQUE_PAGES");
dimension.SetValue("URLS");

Aws::CloudWatch::Model::MetricDatum datum;
datum.SetMetricName("PAGES_VISITED");
datum.SetUnit(Aws::CloudWatch::Model::StandardUnit::None);
datum.SetValue(data_point);
datum.AddDimensions(dimension);

Aws::CloudWatch::Model::PutMetricDataRequest request;
request.SetNamespace("SITE/TRAFFIC");
request.AddMetricData(datum);

auto outcome = cw.PutMetricData(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to put sample metric data:" <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully put sample metric data" << std::endl;
}
```

- For API details, see [PutMetricData](#) in *AWS SDK for C++ API Reference*.

## CloudWatch Logs examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon CloudWatch Logs.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2616\)](#)

## Actions

### Create a subscription filter

The following code example shows how to create an Amazon CloudWatch Logs subscription filter.

#### SDK for C++

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/PutSubscriptionFilterRequest.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Create the subscription filter.

```
Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
Aws::CloudWatchLogs::Model::PutSubscriptionFilterRequest request;
request.SetFilterName(filter_name);
request.SetFilterPattern(filter_pattern);
request.SetLogGroupName(log_group);
request.SetDestinationArn(dest_arn);
auto outcome = cwl.PutSubscriptionFilter(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch logs subscription filter "
        << filter_name << ":" << outcome.GetError().GetMessage() <<
        std::endl;
}
else
{
    std::cout << "Successfully created CloudWatch logs subscription " <<
        "filter " << filter_name << std::endl;
}
```

- For API details, see [PutSubscriptionFilter](#) in [AWS SDK for C++ API Reference](#).

### Delete a subscription filter

The following code example shows how to delete an Amazon CloudWatch Logs subscription filter.

#### SDK for C++

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>
#include <iostream>
```

Delete the subscription filter.

```
Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
Aws::CloudWatchLogs::Model::DeleteSubscriptionFilterRequest request;
request.SetFilterName(filter_name);
request.SetLogGroupName(log_group);

auto outcome = cwl.DeleteSubscriptionFilter(request);
if (!outcome.IsSuccess()) {
    std::cout << "Failed to delete CloudWatch log subscription filter "
        << filter_name << ":" << outcome.GetError().GetMessage() <<
        std::endl;
} else {
    std::cout << "Successfully deleted CloudWatch logs subscription " <<
        "filter " << filter_name << std::endl;
}
```

- For API details, see [DeleteSubscriptionFilter](#) in *AWS SDK for C++ API Reference*.

## Describe existing subscription filters

The following code example shows how to describe Amazon CloudWatch Logs existing subscription filters.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DescribeSubscriptionFiltersRequest.h>
#include <aws/logs/model/DescribeSubscriptionFiltersResult.h>
#include <iostream>
#include <iomanip>
```

List the subscription filters.

```
Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
Aws::CloudWatchLogs::Model::DescribeSubscriptionFiltersRequest request;
request.SetLogGroupName(log_group);
request.SetLimit(1);

bool done = false;
bool header = false;
while (!done) {
    auto outcome = cwl.DescribeSubscriptionFilters(
```

```
    request);
if (!outcome.IsSuccess()) {
    std::cout << "Failed to describe CloudWatch subscription filters "
    << "for log group " << log_group << ":" <<
    outcome.GetError().GetMessage() << std::endl;
    break;
}

if (!header) {
    std::cout << std::left << std::setw(32) << "Name" <<
        std::setw(64) << "FilterPattern" << std::setw(64) <<
        "DestinationArn" << std::endl;
    header = true;
}

const auto &filters = outcome.GetResult().GetSubscriptionFilters();
for (const auto &filter : filters) {
    std::cout << std::left << std::setw(32) <<
        filter.GetFilterName() << std::setw(64) <<
        filter.GetFilterPattern() << std::setw(64) <<
        filter.GetDestinationArn() << std::endl;
}

const auto &next_token = outcome.GetResult().GetNextToken();
request.SetNextToken(next_token);
done = next_token.empty();
}
```

- For API details, see [DescribeSubscriptionFilters](#) in *AWS SDK for C++ API Reference*.

## DynamoDB examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon DynamoDB.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2618\)](#)
- [Scenarios \(p. 2632\)](#)

## Actions

### Create a table

The following code example shows how to create a DynamoDB table.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
if (!region.empty())
```

```
    clientConfig.region = region;
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

std::cout << "Creating table " << table <<
    " with a simple primary key: \"Name\"" << std::endl;

Aws::DynamoDB::Model::CreateTableRequest req;

Aws::DynamoDB::Model::AttributeDefinition haskKey;
haskKey.SetAttributeName("Name");
haskKey.SetAttributeType(Aws::DynamoDB::Model::ScalarAttributeType::S);
req.AddAttributeDefinitions(haskKey);

Aws::DynamoDB::Model::KeySchemaElement keyselt;

keyselt.WithAttributeName("Name").WithKeyType(Aws::DynamoDB::Model::KeyType::HASH);
req.AddKeySchema(keyselt);

Aws::DynamoDB::Model::ProvisionedThroughput thruput;
thruput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
req.SetProvisionedThroughput(thruput);
req.SetTableName(table);

const Aws::DynamoDB::Model::CreateTableOutcome& result =
dynamoClient.CreateTable(req);
if (result.IsSuccess())
{
    std::cout << "Table \\" <<
result.GetResult().GetTableDescription().GetTableName() <<
        " created!" << std::endl;
}
else
{
    std::cout << "Failed to create table: " << result.GetError().GetMessage();
}
```

- For API details, see [CreateTable](#) in *AWS SDK for C++ API Reference*.

## Delete a table

The following code example shows how to delete a DynamoDB table.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
if (!region.empty())
    clientConfig.region = region;
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

Aws::DynamoDB::Model::DeleteTableRequest dtr;
dtr.SetTableName(table);

const Aws::DynamoDB::Model::DeleteTableOutcome& result =
dynamoClient.DeleteTable(dtr);
if (result.IsSuccess())
{
    std::cout << "Your Table \\" <<
result.GetResult().GetTableDescription().GetTableName() << " was deleted!\n";
```

```
        }
    else
    {
        std::cout << "Failed to delete table: " << result.GetError().GetMessage();
    }
}
```

- For API details, see [DeleteTable](#) in *AWS SDK for C++ API Reference*.

## Get an item from a table

The following code example shows how to get an item from a DynamoDB table.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);
Aws::DynamoDB::Model::GetItemRequest req;

// Set up the request.
req.SetTableName(table);
Aws::DynamoDB::Model::AttributeValue hashKey;
hashKey.SetS(keyval);
req.AddKey(key, hashKey);

// Retrieve the item's fields and values
const Aws::DynamoDB::Model::GetItemOutcome& result = dynamoClient.GetItem(req);
if (result.IsSuccess())
{
    // Reference the retrieved fields/values.
    const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>& item =
result.GetResult().GetItem();
    if (item.size() > 0)
    {
        // Output each retrieved field and its value.
        for (const auto& i : item)
            std::cout << "Values: " << i.first << ":" << i.second.GetS() <<
std::endl;
    }
    else
    {
        std::cout << "No item found with the key " << key << std::endl;
    }
}
else
{
    std::cout << "Failed to get item: " << result.GetError().GetMessage();
}
```

- For API details, see [GetItem](#) in *AWS SDK for C++ API Reference*.

## Get information about a table

The following code example shows how to get information about a DynamoDB table.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
if (!region.empty())
    clientConfig.region = region;
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

Aws::DynamoDB::Model::DescribeTableRequest dtr;
dtr.SetTableName(table);

const Aws::DynamoDB::Model::DescribeTableOutcome& result =
dynamoClient.DescribeTable(dtr);

if (result.IsSuccess())
{
    const Aws::DynamoDB::Model::TableDescription& td =
result.GetResult().GetTable();
    std::cout << "Table name : " << td.GetTableName() << std::endl;
    std::cout << "Table ARN : " << td.GetTableArn() << std::endl;
    std::cout << "Status : " <<
Aws::DynamoDB::Model::TableStatusMapper::GetNameForTableStatus(td.GetTableStatus()) <<
std::endl;
    std::cout << "Item count : " << td.GetItemCount() << std::endl;
    std::cout << "Size (bytes): " << td.GetTableSizeBytes() << std::endl;

    const Aws::DynamoDB::Model::ProvisionedThroughputDescription& ptd =
td.GetProvisionedThroughput();
    std::cout << "Throughput" << std::endl;
    std::cout << " Read Capacity : " << ptd.GetReadCapacityUnits() <<
std::endl;
    std::cout << " Write Capacity: " << ptd.GetWriteCapacityUnits() <<
std::endl;

    const Aws::Vector<Aws::DynamoDB::Model::AttributeDefinition>& ad =
td.GetAttributeDefinitions();
    std::cout << "Attributes" << std::endl;
    for (const auto& a : ad)
        std::cout << " " << a.getAttributeName() << "(" <<

Aws::DynamoDB::Model::ScalarAttributeTypeMapper::GetNameForScalarAttributeType(a.getAttributeType()
<<
")" << std::endl;
}
else
{
    std::cout << "Failed to describe table: " <<
result.GetError().GetMessage();
}
```

- For API details, see [DescribeTable](#) in *AWS SDK for C++ API Reference*.

## List tables

The following code example shows how to list DynamoDB tables.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
listTablesRequest.SetLimit(50);
do
{
    const Aws::DynamoDB::Model::ListTablesOutcome& lto =
dynamoClient.ListTables(listTablesRequest);
    if (!lto.IsSuccess())
    {
        std::cout << "Error: " << lto.GetError().GetMessage() << std::endl;
        return 1;
    }

    for (const auto& s : lto.GetResult().GetTableNames())
        std::cout << s << std::endl;

listTablesRequest.SetExclusiveStartTableName(lto.GetResult().GetLastEvaluatedTableName());
} while (!listTablesRequest.GetExclusiveStartTableName().empty());
```

- For API details, see [ListTables](#) in *AWS SDK for C++ API Reference*.

## Put an item in a table

The following code example shows how to put an item in a DynamoDB table.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

Aws::DynamoDB::Model::PutItemRequest putItemRequest;
putItemRequest.SetTableName(table);

Aws::DynamoDB::Model::AttributeValue av;
av.SetS(keyVal);

Aws::DynamoDB::Model::AttributeValue album;
album.SetS(AlbumTitleValue);

Aws::DynamoDB::Model::AttributeValue awards;
awards.SetS(AwardVal);

Aws::DynamoDB::Model::AttributeValue song;
song.SetS(SongTitleVal);

// Add all AttributeValue objects.
```

```
putItemRequest.AddItem(key, av);
putItemRequest.AddItem(albumTitle, album);
putItemRequest.AddItem(Awards, awards);
putItemRequest.AddItem(SongTitle, song);

const Aws::DynamoDB::Model::PutItemOutcome result =
dynamoClient.PutItem(putItemRequest);
if (!result.IsSuccess())
{
    std::cout << result.GetError().GetMessage() << std::endl;
    return 1;
}
std::cout << "Successfully added Item!" << std::endl;
```

- For API details, see [PutItem](#) in *AWS SDK for C++ API Reference*.

## Query a table

The following code example shows how to query a DynamoDB table.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;

Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);
Aws::DynamoDB::Model::QueryRequest req;

req.SetTableName(table);

// Set query key condition expression
req.SetKeyConditionExpression(partitionKeyAttributeName + "= :valueToMatch");

// Set Expression AttributeValues
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> attributeValues;
attributeValues.emplace(":valueToMatch", partitionKeyValue);

req.SetExpressionAttributeValues(attributeValues);

// Perform Query operation
const Aws::DynamoDB::Model::QueryOutcome& result = dynamoClient.Query(req);
if (result.IsSuccess())
{
    // Reference the retrieved items
    const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>>& items = result.GetResult().GetItems();
    if(items.size() > 0)
    {
        std::cout << "Number of items retrieved from Query: " << items.size()
<< std::endl;
        //Iterate each item and print
        for(const auto &item: items)
        {
            std::cout << "*****";
        }
        std::cout << std::endl;
        // Output each retrieved field and its value
        for (const auto& i : item)
            std::cout << i.first << ":" << i.second.GetS() << std::endl;
    }
}
```

```
        }
    }

    else
    {
        std::cout << "No item found in table: " << table << std::endl;
    }
else
{
    std::cout << "Failed to Query items: " << result.GetError().GetMessage();
}
```

- For API details, see [Query in AWS SDK for C++ API Reference](#).

## Run a PartiQL statement

The following code example shows how to run a PartiQL statement on a DynamoDB table.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use an INSERT statement to add an item.

```
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

// 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
Aws::String title;
float rating;
int year;
Aws::String plot;
{
    title = askQuestion(
        "Enter the title of a movie you want to add to the table: ");
    year = askQuestionForInt("What year was it released? ");
    rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it? ",
        1, 10);
    plot = askQuestion("Summarize the plot for me: ");

    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "INSERT INTO \"<< MOVIE_TABLE_NAME << \" VALUE {"
        << TITLE_KEY << ": ?, \" << YEAR_KEY << ": ?, "
        << INFO_KEY << ": ?}";

    request.SetStatement(sqlStream.str());

    // Create the parameter attributes.
    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

    Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

    std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
    Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
        ALLOCATION_TAG.c_str());
    ratingAttribute->SetN(rating);
    infoMapAttribute.AddMEEntry(RATING_KEY, ratingAttribute);
```

```
    std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
    ALLOCATION_TAG.c_str());
plotAttribute->SetS(plot);
infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
attributes.push_back(infoMapAttribute);
request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to add a movie: " << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
}
```

Use a SELECT statement to get an item.

```
// 3. Get the data for the movie using a "Select" statement. (ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM \\" " << MOVIE_TABLE_NAME << "\\ WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to retrieve movie information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    else {
        // Print the retrieved movie information.
        const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

        if (items.size() == 1) {
            printMovieInfo(items[0]);
        }
        else {
            std::cerr << "Error: " << items.size() << " movies were retrieved. "
                << " There should be only one movie." << std::endl;
        }
    }
}
```

Use an UPDATE statement to update an item.

```
// 4. Update the data for the movie using an "Update" statement.  
(ExecuteStatement)  
{  
    rating = askQuestionForFloatRange(  
        Aws::String("\nLet's update your movie.\nYou rated it  ") +  
        std::to_string(rating)  
        + ", what new rating would you give it? ", 1, 10);  
  
    Aws::DynamoDB::Model::ExecuteStatementRequest request;  
    std::stringstream sqlStream;  
    sqlStream << "UPDATE \"<< MOVIE_TABLE_NAME << \" SET "  
    << INFO_KEY << "." << RATING_KEY << "=? WHERE "  
    << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";  
  
    request.SetStatement(sqlStream.str());  
  
    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;  
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));  
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));  
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));  
  
    request.SetParameters(attributes);  
  
    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =  
    dynamoClient.ExecuteStatement(  
        request);  
  
    if (!outcome.IsSuccess()) {  
        std::cerr << "Failed to update a movie: "  
        << outcome.GetError().GetMessage();  
        return false;  
    }  
}
```

Use a DELETE statement to delete an item.

```
// 6. Delete the movie using a "Delete" statement. (ExecuteStatement)  
{  
    Aws::DynamoDB::Model::ExecuteStatementRequest request;  
    std::stringstream sqlStream;  
    sqlStream << "DELETE FROM \"<< MOVIE_TABLE_NAME << \" WHERE "  
    << TITLE_KEY << "=? and " << YEAR_KEY << "=?";  
  
    request.SetStatement(sqlStream.str());  
  
    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;  
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));  
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));  
    request.SetParameters(attributes);  
  
    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =  
    dynamoClient.ExecuteStatement(  
        request);  
    if (!outcome.IsSuccess()) {  
        std::cerr << "Failed to delete the movie: "  
        << outcome.GetError().GetMessage() << std::endl;  
        return false;  
    }  
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for C++ API Reference*.

## Run batches of PartiQL statements

The following code example shows how to run batches of PartiQL statements on a DynamoDB table.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use batches of INSERT statements to add items.

```
// 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

std::vector<Aws::String> titles;
std::vector<float> ratings;
std::vector<int> years;
std::vector<Aws::String> plots;
Aws::String doAgain = "n";
do {
    Aws::String aTitle = askQuestion(
        "Enter the title of a movie you want to add to the table: ");
    titles.push_back(aTitle);
    int aYear = askQuestionForInt("What year was it released? ");
    years.push_back(aYear);
    float aRating = askQuestionForFloatRange(
        "On a scale of 1 - 10, how do you rate it? ",
        1, 10);
    ratings.push_back(aRating);
    Aws::String aPlot = askQuestion("Summarize the plot for me: ");
    plots.push_back(aPlot);

    doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/n)"));
} while (doAgain == "y");

std::cout << "Adding " << titles.size()
    << (titles.size() == 1 ? " movie " : " movies ")
    << "to the table using a batch \"INSERT\" statement." << std::endl;

{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());

    std::stringstream sqlStream;
    sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {"
        << TITLE_KEY << ": ?, " << YEAR_KEY << ": ?, "
        << INFO_KEY << ": ?}";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

        // Create attribute for the info map.
```

```

Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

    std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
        ALLOCATION_TAG.c_str());
    ratingAttribute->SetN(ratings[i]);
    infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

    std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
        ALLOCATION_TAG.c_str());
    plotAttribute->SetS(plots[i]);
    infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
    attributes.push_back(infoMapAttribute);
    statements[i].SetParameters(attributes);
}

Aws::DynamoDB::Model::BatchExecuteStatementRequest request;
request.SetStatements(statements);

Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
    request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to add the movies: " <<
outcome.GetError().GetMessage()
        << std::endl;
    return false;
}
}

```

```
// 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM `"
        << MOVIE_TABLE_NAME << "` WHERE "
        << TITLE_KEY << "=?" and " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;
    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
        request);
    if (outcome.IsSuccess()) {
```

```
    const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

    const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse> &responses
= result.GetResponses();

    for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
        const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> &item
= response.GetItem();

        printMovieInfo(item);
    }
} else {
    std::cerr << "Failed to retrieve the movie information: "
        << outcome.GetError().GetMessage() << std::endl;
    return false;
}
}
```

Use batches of UPDATE statements to update items.

```
// 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

for (size_t i = 0; i < titles.size(); ++i) {
    ratings[i] = askQuestionForFloatRange(
        Aws::String("\nLet's update your the movie, \"") + titles[i] +
        ".\nYou rated it " + std::to_string(ratings[i]) +
        ", what new rating would you give it? ", 1, 10);
}

std::cout << "Updating the movie with a batch \"UPDATE\" statement." << std::endl;

{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());

    std::stringstream sqlStream;
    sqlStream << "UPDATE \"<< MOVIE_TABLE_NAME << \" SET "
        << INFO_KEY << "." << RATING_KEY << "=? WHERE "
        << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;
    request.SetStatements(statements);
```

```
Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
    request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to update movie information: "
        << outcome.GetError().GetMessage() << std::endl;
    return false;
}
```

Use batches of DELETE statements to delete items.

```
// 6. Delete multiple movies using "Delete" statements. (BatchExecuteStatement)
{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "DELETE FROM `"
        << MOVIE_TABLE_NAME << "` WHERE "
        << TITLE_KEY << "=?" and " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete the movies: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for C++ API Reference*.

## Scan a table

The following code example shows how to scan a DynamoDB table.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
```

```
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);
Aws::DynamoDB::Model::ScanRequest req;
req.SetTableName(table);

if (!projection.empty())
    req.SetProjectionExpression(projection);

// Perform scan on table
const Aws::DynamoDB::Model::ScanOutcome& result = dynamoClient.Scan(req);
if (result.IsSuccess())
{
    // Reference the retrieved items
    const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>>& items = result.GetResult().GetItems();
    if(items.size() > 0)
    {
        std::cout << "Number of items retrieved from scan: " << items.size() <<
std::endl;
        //Iterate each item and print
        for(const auto &item: items)
        {
            std::cout << "*****";
<< std::endl;
            // Output each retrieved field and its value
            for (const auto& i : item)
                std::cout << i.first << ":" << i.second.GetS() << std::endl;
        }
    }

    else
    {
        std::cout << "No item found in table: " << table << std::endl;
    }
}
else
{
    std::cout << "Failed to Scan items: " << result.GetError().GetMessage();
}
```

- For API details, see [Scan](#) in *AWS SDK for C++ API Reference*.

## Update an item in a table

The following code example shows how to update an item in a DynamoDB table.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfig);

// *** Define UpdateItem request arguments
// Define TableName argument.
Aws::DynamoDB::Model::UpdateItemRequest request;
request.SetTableName(tableName);

// Define KeyName argument.
```

```
Aws::DynamoDB::Model::AttributeValue attribValue;
attribValue.SetS(keyValue);
request.AddKey("id", attribValue);

// Construct the SET update expression argument.
Aws::String update_expression("SET #a = :valueA");
request.SetUpdateExpression(update_expression);

// Parse the attribute name and value. Syntax: "name=value".
auto parsed = Aws::Utils::StringUtils::Split(attributeNameAndValue, '=');

if (parsed.size() != 2)
{
    std::cout << "Invalid argument syntax: " << attributeNameAndValue << USAGE;
    return 1;
}

// Construct attribute name argument
// Note: Setting the ExpressionAttributeNames argument is required only
// when the name is a reserved word, such as "default". Otherwise, the
// name can be included in the update_expression, as in
// "SET MyAttributeName = :valueA"
Aws::Map<Aws::String, Aws::String> expressionAttributeNames;
expressionAttributeNames["#a"] = parsed[0];
request.SetExpressionAttributeNames(expressionAttributeNames);

// Construct attribute value argument.
Aws::DynamoDB::Model::AttributeValue attributeUpdatedValue;
attributeUpdatedValue.Sets(parsed[1]);
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
expressionAttributeValues;
expressionAttributeValues[":valueA"] = attributeUpdatedValue;
request.SetExpressionAttributeValues(expressionAttributeValues);

// Update the item.
const Aws::DynamoDB::Model::UpdateItemOutcome& result =
dynamoClient.UpdateItem(request);
if (!result.IsSuccess())
{
    std::cout << result.GetError().GetMessage() << std::endl;
    return 1;
}
std::cout << "Item was updated" << std::endl;
```

- For API details, see [UpdateItem](#) in *AWS SDK for C++ API Reference*.

## Scenarios

### Get started using tables, items, and queries

The following code example shows how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.
- Delete a movie from the table.
- Delete the table.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
{  
    Aws::Client::ClientConfiguration clientConfig;  
    // 1. Create a table with partition: year (N) and sort: title (S).  
(CreateTable)  
    if (AwsDoc::DynamoDB::createDynamoDBTable(AwsDoc::DynamoDB::MOVIE_TABLE_NAME,  
                                                clientConfig)) {  
  
        AwsDoc::DynamoDB::dynamodbGettingStartedScenario(clientConfig);  
  
        // 9. Delete the table. (DeleteTable)  
        AwsDoc::DynamoDB::deleteDynamoTable(AwsDoc::DynamoDB::MOVIE_TABLE_NAME,  
                                              clientConfig);  
    }  
}  
  
//! Scenario to modify and query a DynamoDB table.  
/*!  
 \sa dynamodbGettingStartedScenario()  
 \param clientConfiguration: Aws client configuration.  
 \return bool: Function succeeded.  
 */  
bool AwsDoc::DynamoDB::dynamodbGettingStartedScenario(  
    const Aws::Client::ClientConfiguration &clientConfiguration) {  
    std::cout << std::setfill('*') << std::setw(ASTERIX_FILL_WIDTH) << " " <<  
    std::endl;  
    std::cout << "Welcome to the Amazon DynamoDB getting started demo." << std::endl;  
    std::cout << std::setfill('*') << std::setw(ASTERIX_FILL_WIDTH) << " " <<  
    std::endl;  
  
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);  
  
    // 2. Add a new movie.  
    Aws::String title;  
    float rating;  
    int year;  
    Aws::String plot;  
    {  
        title = askQuestion(  
            "Enter the title of a movie you want to add to the table: ");  
        year = askQuestionForInt("What year was it released? ");  
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it? ",  
                                         1, 10);  
        plot = askQuestion("Summarize the plot for me: ");  
  
        Aws::DynamoDB::Model::PutItemRequest putItemRequest;  
        putItemRequest.SetTableName(MOVIE_TABLE_NAME);  
  
        putItemRequest.AddItem(YEAR_KEY,  
                               Aws::DynamoDB::Model::AttributeValue().SetN(year));  
        putItemRequest.AddItem(TITLE_KEY,  
                               Aws::DynamoDB::Model::AttributeValue().SetS(title));  
  
        // Create attribute for the info map.  
        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;  
  
        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =  
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(  
            ALLOCATION_TAG.c_str());
```

```

ratingAttribute->SetN(rating);
infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
    ALLOCATION_TAG.c_str());
plotAttribute->SetS(plot);
infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);

putItemRequest.AddItem(INFO_KEY, infoMapAttribute);

Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
    putItemRequest);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to add an item: " << outcome.GetError().GetMessage()
        << std::endl;
    return false;
}
}

std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
    << std::endl;

// 3. Update the rating and plot of the movie by using an update expression.
{
    rating = askQuestionForFloatRange(
        Aws::String("\nLet's update your movie.\nYou rated it  ") +
        std::to_string(rating)
        + ", what new rating would you give it? ", 1, 10);
    plot = askQuestion(Aws::String("You summarized the plot as ''") + plot +
        "'.\nWhat would you say now? ");

    Aws::DynamoDB::Model::UpdateItemRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);
    request.AddKey(TITLE_KEY, Aws::DynamoDB::Model::AttributeValue().SetS(title));
    request.AddKey(YEAR_KEY, Aws::DynamoDB::Model::AttributeValue().SetN(year));
    std::stringstream expressionStream;
    expressionStream << "set " << INFO_KEY << "." << RATING_KEY << " :=:r, "
        << INFO_KEY << "." << PLOT_KEY << " :=:p";
    request.SetUpdateExpression(expressionStream.str());
    request.SetExpressionAttributeValues({
        {"":r",
        Aws::DynamoDB::Model::AttributeValue().SetN(
            rating)},
        {"":p",
        Aws::DynamoDB::Model::AttributeValue().SetS(
            plot)}
    });

    request.SetReturnValues(Aws::DynamoDB::Model::ReturnValue::UPDATED_NEW);

    const Aws::DynamoDB::Model::UpdateItemOutcome &result =
dynamoClient.UpdateItem(
    request);
    if (!result.IsSuccess()) {
        std::cerr << "Error updating movie " + result.GetError().GetMessage()
            << std::endl;
        return false;
    }
}

std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;

// 4. Put 250 movies in the table from moviedata.json.
{
    std::cout << "Adding movies from a json file to the database." << std::endl;
}

```

```

const size_t MAX_SIZE_FOR_BATCH_WRITE = 25;
const size_t MOVIES_TO_WRITE = 10 * MAX_SIZE_FOR_BATCH_WRITE;
Aws::String jsonString = getMovieJSON();
if (!jsonString.empty()) {
    Aws::Utils::JsonValue json(jsonString);
    Aws::Utils::Array<Aws::Utils::JsonValue> movieJsons =
    json.View().AsArray();
    Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;

    // To add movies with a cross-section of years, use an appropriate
increment
    // value for iterating through the database.
    size_t increment = movieJsons.GetLength() / MOVIES_TO_WRITE;
    for (size_t i = 0; i < movieJsons.GetLength(); i += increment) {
        writeRequests.push_back(Aws::DynamoDB::Model::WriteRequest());
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> putItems =
        movieJsonViewToAttributeMap(
            movieJsons[i]);
        Aws::DynamoDB::Model::PutRequest putRequest;
        putRequest.SetItem(putItems);
        writeRequests.back().SetPutRequest(putRequest);
        if (writeRequests.size() == MAX_SIZE_FOR_BATCH_WRITE) {
            Aws::DynamoDB::Model::BatchWriteItemRequest request;
            request.AddRequestItems(MOVIE_TABLE_NAME, writeRequests);
            const Aws::DynamoDB::Model::BatchWriteItemOutcome &outcome =
            dynamoClient.BatchWriteItem(
                request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Unable to batch write movie data: "
                << outcome.GetError().GetMessage()
                << std::endl;
                writeRequests.clear();
                break;
            }
            else {
                std::cout << "Added batch of " << writeRequests.size()
                << " movies to the database."
                << std::endl;
            }
            writeRequests.clear();
        }
    }
}

std::cout << std::setfill('*') << std::setw(ASTERIX_FILL_WIDTH) << " " <<
std::endl;

// 5. Get a movie by Key (partition + sort).
{
    Aws::String titleToGet("King Kong");
    Aws::String answer = askQuestion(Aws::String(
        "Let's move on...Would you like to get info about '" + titleToGet +
        "? (y/n) "));
    if (answer == "y") {
        Aws::DynamoDB::Model::GetItemRequest request;
        request.SetTableName(MOVIE_TABLE_NAME);
        request.AddKey(TITLE_KEY,
            Aws::DynamoDB::Model::AttributeValue().SetS(titleToGet));
        request.AddKey(YEAR_KEY,
            Aws::DynamoDB::Model::AttributeValue().SetN(1933));

        const Aws::DynamoDB::Model::GetItemOutcome &result = dynamoClient.GetItem(
            request);
        if (!result.IsSuccess()) {
            std::cerr << "Error " << result.GetError().GetMessage();
        }
    }
}

```

```

        }
        else {
            const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> &item
= result.GetResult().GetItem();
            if (!item.empty()) {
                std::cout << "\nHere's what I found:" << std::endl;
                printMovieInfo(item);
            }
            else {
                std::cout << "\nThe movie was not found in the database."
                    << std::endl;
            }
        }
    }

// 6. Use Query with a key condition expression to return all movies
//     released in a given year.
Aws::String doAgain = "n";
do {
    Aws::DynamoDB::Model::QueryRequest req;

    req.SetTableName(MOVIE_TABLE_NAME);

    // "year" is a DynamoDB reserved keyword and must be replaced with an
    // expression attribute name.
    req.SetKeyConditionExpression("#dynobase_year = :valueToMatch");
    req.SetExpressionAttributeNames({{"#dynobase_year", YEAR_KEY}});

    int yearToMatch = askQuestionForIntRange(
        "\nLet's get a list of movies released in"
        " a given year. Enter a year between 1972 and 2018 ",
        1972, 2018);
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> attributeValues;
    attributeValues.emplace(":valueToMatch",
                           Aws::DynamoDB::Model::AttributeValue().SetN(
                               yearToMatch));
    req.SetExpressionAttributeValues(attributeValues);

    const Aws::DynamoDB::Model::QueryOutcome &result = dynamoClient.Query(req);
    if (result.IsSuccess()) {
        const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
        if (!items.empty()) {
            std::cout << "\nThere were " << items.size()
                << " movies in the database from "
                << yearToMatch << "." << std::endl;
            for (const auto &item: items) {
                printMovieInfo(item);
            }
            doAgain = "n";
        }
        else {
            std::cout << "\nNo movies from " << yearToMatch
                << " were found in the database"
                << std::endl;
            doAgain = askQuestion(Aws::String("Try another year? (y/n) "));
        }
    }
    else {
        std::cerr << "Failed to Query items: " << result.GetError().GetMessage()
            << std::endl;
    }
}

} while (doAgain == "y");

```

```

// 7. Use Scan to return movies released within a range of years.
// Show how to paginate data using ExclusiveStartKey. (Scan + FilterExpression)
{
    int startYear = askQuestionForIntRange("\nNow let's scan a range of years "
                                         "for movies in the database. Enter a
start year: ",
                                         1972, 2018);
    int endYear = askQuestionForIntRange("\nEnter an end year: ",
                                         startYear, 2018);
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> exclusiveStartKey;
do {
    Aws::DynamoDB::Model::ScanRequest scanRequest;
    scanRequest.SetTableName(MOVIE_TABLE_NAME);
    scanRequest.SetFilterExpression(
        "#dynobase_year >= :startYear AND #dynobase_year <= :endYear");
    scanRequest.SetExpressionAttributeNames({{"#dynobase_year", YEAR_KEY}});

    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributeValues;
    attributeValues.emplace(":startYear",
                           Aws::DynamoDB::Model::AttributeValue().SetN(
                               startYear));
    attributeValues.emplace(":endYear",
                           Aws::DynamoDB::Model::AttributeValue().SetN(
                               endYear));
    scanRequest.SetExpressionAttributeValues(attributeValues);

    if (!exclusiveStartKey.empty()) {
        scanRequest.SetExclusiveStartKey(exclusiveStartKey);
    }

    const Aws::DynamoDB::Model::ScanOutcome &result = dynamoClient.Scan(
        scanRequest);
    if (result.IsSuccess()) {
        const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
        if (!items.empty()) {
            std::stringstream stringStream;
            stringStream << "\nFound " << items.size() << " movies in one
scan."
            << " How many would you like to see? ";
            size_t count = askQuestionForInt(stringStream.str());
            for (size_t i = 0; i < count && i < items.size(); ++i) {
                printMovieInfo(items[i]);
            }
        }
        else {
            std::cout << "\nNo movies in the database between " << startYear <<
            " and " << endYear << "." << std::endl;
        }
    }

    exclusiveStartKey = result.GetResult().GetLastEvaluatedKey();
    if (!exclusiveStartKey.empty()) {
        std::cout << "Not all movies were retrieved. Scanning for more."
             << std::endl;
    }
    else {
        std::cout << "All movies were retrieved with this scan."
             << std::endl;
    }
}
else {
    std::cerr << "Failed to Scan movies: "
             << result.GetError().GetMessage() << std::endl;
}
} while (!exclusiveStartKey.empty());

```

```

        }

        // 8. Delete a movie. (DeleteItem)
        {
            std::stringstream stringStream;
            stringStream << "\nWould you like to delete the movie " << title
                << " from the database? (y/n) ";
            Aws::String answer = askQuestion(stringStream.str());
            if (answer == "y") {
                Aws::DynamoDB::Model::DeleteItemRequest request;
                request.AddKey(YEAR_KEY,
                    Aws::DynamoDB::Model::AttributeValue().SetN(year));
                request.AddKey(TITLE_KEY,
                    Aws::DynamoDB::Model::AttributeValue().SetS(title));
                request.SetTableName(MOVIE_TABLE_NAME);

                const Aws::DynamoDB::Model::DeleteItemOutcome &result =
dynamoClient.DeleteItem(
                    request);
                if (result.IsSuccess()) {
                    std::cout << "\nRemoved '" << title << "' from the database."
                        << std::endl;
                }
                else {
                    std::cerr << "Failed to delete the movie: "
                        << result.GetError().GetMessage()
                        << std::endl;
                }
            }
        }

        return true;
    }

    //! Routine to convert a JsonView object to an attribute map.
/*!
\sa movieJsonViewToAttributeMap()
\param jsonView: Json view object.
\return map: Map that can be used in a DynamoDB request.
*/
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
AwsDoc::DynamoDB::movieJsonViewToAttributeMap(
    const Aws::Utils::Json::JsonView &jsonView) {
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> result;

    if (jsonView.KeyExists(YEAR_KEY)) {
        result[YEAR_KEY].SetN(jsonView.GetInteger(YEAR_KEY));
    }
    if (jsonView.KeyExists(TITLE_KEY)) {
        result[TITLE_KEY].SetS(jsonView.GetString(TITLE_KEY));
    }
    if (jsonView.KeyExists(INFO_KEY)) {
        Aws::Map<Aws::String, const
std::shared_ptr<Aws::DynamoDB::Model::AttributeValue>> infoMap;
        Aws::Utils::Json::JsonView infoView = jsonView.GetObject(INFO_KEY);
        if (infoView.KeyExists(RATING_KEY)) {
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue =
std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
            attributeValue->SetN(infoView.GetDouble(RATING_KEY));
            infoMap.emplace(std::make_pair(RATING_KEY, attributeValue));
        }
        if (infoView.KeyExists(PLOT_KEY)) {
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue =
std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
            attributeValue->SetS(infoView.GetString(PLOT_KEY));
            infoMap.emplace(std::make_pair(PLOT_KEY, attributeValue));
        }
    }
}

```

```

        }

        result[INFO_KEY].SetM(infoMap);
    }

    return result;
}

/// Create a DynamoDB table.
*/
\sa createDynamoDBTable()
\param tableName: The DynamoDB table's name.
\param clientConfiguration: Aws client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createDynamoDBTable(const Aws::String &tableName,
                                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
            Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
            Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(titleAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
            Aws::DynamoDB::Model::KeyType::HASH);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
        yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
            Aws::DynamoDB::Model::KeyType::RANGE);
        request.AddKeySchema(titleKeySchema);

        Aws::DynamoDB::Model::ProvisionedThroughput throughput;
        throughput.WithReadCapacityUnits(
            PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
            PROVISIONED_THROUGHPUT_UNITS);
        request.SetProvisionedThroughput(throughput);
        request.SetTableName(MOVIE_TABLE_NAME);

        std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
        const Aws::DynamoDB::Model::CreateTableOutcome &result =
        dynamoClient.CreateTable(
            request);
        if (!result.IsSuccess()) {
            if (result.GetError().GetErrorCode() ==
                Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
                std::cout << "Table already exists." << std::endl;
                movieTableAlreadyExisted = true;
            }
            else {
                std::cerr << "Failed to create table: "

```

```

                << result.GetError().GetMessage();
            return false;
        }
    }

    // Wait for table to become active.
    if (!movieTableAlreadyExisted) {
        std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
        << "' to become active...." << std::endl;
        if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME, clientConfiguration))
    {
        return false;
    }
    std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
    << std::endl;
}

return true;
}

/// Delete a DynamoDB table.
*/
\sa deleteDynamoTable()
\param tableName: The DynamoDB table's name.
\param clientConfiguration: Aws client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteDynamoTable(const Aws::String &tableName,
                                            const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result = dynamoClient.DeleteTable(
        request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
        << result.GetResult().GetTableDescription().GetTableName()
        << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
        << std::endl;
    }
}

return result.IsSuccess();
}

/// Query a newly created DynamoDB table until it is active.
*/
\sa waitTableActive()
\param waitTableActive: The DynamoDB table's name.
\param clientConfiguration: Aws client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                         const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);
}

```

```
int count = 0;
while (count < MAX_QUERIES) {
    const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
    request);
    if (result.IsSuccess()) {
        Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

        if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {
            return true;
        }
    }
    else {
        std::cerr << "Error DynamoDB::waitTableActive "
             << result.GetError().GetMessage() << std::endl;
        return false;
    }
    count++;
}
return false;
}
```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## Query a table by using batches of PartiQL statements

The following code example shows how to:

- Get a batch of items by running multiple SELECT statements.
- Add a batch of items by running multiple INSERT statements.
- Update a batch of items by running multiple UPDATE statements.
- Delete a batch of items by running multiple DELETE statements.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::Client::ClientConfiguration clientConfig;
```

```

// 1. Create a table. (CreateTable)
if (AwsDoc::DynamoDB::createDynamoDBTable(AwsDoc::DynamoDB::MOVIE_TABLE_NAME,
                                             clientConfig)) {

    AwsDoc::DynamoDB::partiqlBatchExecuteScenario(clientConfig);

    // 7. Delete the table. (DeleteTable)
    AwsDoc::DynamoDB::deleteDynamoTable(AwsDoc::DynamoDB::MOVIE_TABLE_NAME,
                                         clientConfig);
}

/// Scenario to modify and query a DynamoDB table using PartiQL batch statements.
/*!
 \sa partiqlBatchExecuteScenario()
 \param clientConfiguration: Aws client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::partiqlBatchExecuteScenario(
    const Aws::Client::ClientConfiguration &clientConfiguration) {

    // 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::vector<Aws::String> titles;
    std::vector<float> ratings;
    std::vector<int> years;
    std::vector<Aws::String> plots;
    Aws::String doAgain = "n";
    do {
        Aws::String aTitle = askQuestion(
            "Enter the title of a movie you want to add to the table: ");
        titles.push_back(aTitle);
        int aYear = askQuestionForInt("What year was it released? ");
        years.push_back(aYear);
        float aRating = askQuestionForFloatRange(
            "On a scale of 1 - 10, how do you rate it? ",
            1, 10);
        ratings.push_back(aRating);
        Aws::String aPlot = askQuestion("Summarize the plot for me: ");
        plots.push_back(aPlot);

        doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/n")
        ));
    } while (doAgain == "y");

    std::cout << "Adding " << titles.size()
        << (titles.size() == 1 ? " movie " : " movies ")
        << "to the table using a batch \"INSERT\" statement." << std::endl;
}

{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());

    std::stringstream sqlStream;
    sqlStream << "INSERT INTO \""
        << MOVIE_TABLE_NAME << "\" VALUE {"
        << TITLE_KEY << ": ?, \" << YEAR_KEY << ": ?, "
        << INFO_KEY << ": ?}";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
    }
}

```

```

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

        // Create attribute for the info map.
        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(ratings[i]);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plots[i]);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
        attributes.push_back(infoMapAttribute);
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to add the movies: " <<
outcome.GetError().GetMessage()
                << std::endl;
        return false;
    }
}

std::cout << "Retrieving the movie data with a batch \"SELECT\" statement."
    << std::endl;

// 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM " << MOVIE_TABLE_NAME << "\ WHERE "
        << TITLE_KEY << "=?" and " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(

```

```

        request);
    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse> &responses
= result.GetResponses();

        for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
            const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> &item
= response.GetItem();

            printMovieInfo(item);
        }
    } else {
        std::cerr << "Failed to retrieve the movie information: "
                << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

// 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

for (size_t i = 0; i < titles.size(); ++i) {
    ratings[i] = askQuestionForFloatRange(
        Aws::String("\nLet's update your the movie, \"") + titles[i] +
        ".\nYou rated it " + std::to_string(ratings[i]) +
        ", what new rating would you give it? ", 1, 10);
}

std::cout << "Updating the movie with a batch \"UPDATE\" statement." << std::endl;

{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
titles.size());

    std::stringstream sqlStream;
    sqlStream << "UPDATE \"<< MOVIE_TABLE_NAME << \" SET "
              << INFO_KEY << "." << RATING_KEY << "=? WHERE "
              << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);
    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
        request);
}

```

```
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update movie information: "
                << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    std::cout << "Retrieving the updated movie data with a batch \"SELECT\" statement."
        << std::endl;

    // 5. Get the updated data for multiple movies using "Select" statements.
    (BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
            titles.size());
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM \\" " << MOVIE_TABLE_NAME << "\\" WHERE "
            << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;
        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
            request);
        if (outcome.IsSuccess()) {
            const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

            const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse> &responses
= result.GetResponses();

            for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
                const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> &item
= response.GetItem();

                printMovieInfo(item);
            }
        }
        else {
            std::cerr << "Failed to retrieve the movies information: "
                << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    std::cout << "Deleting the movie data with a batch \"DELETE\" statement."
        << std::endl;

    // 6. Delete multiple movies using "Delete" statements. (BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
```

```
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "DELETE FROM `\" " << MOVIE_TABLE_NAME << "`\" WHERE "
    << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete the movies: "
                << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

return true;
}

//! Create a DynamoDB table.
/*!
\sa createDynamoDBTable()
\param tableName: The DynamoDB table's name.
\param clientConfiguration: Aws client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createDynamoDBTable(const Aws::String &tableName,
                                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
            Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
            Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(titleAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
```

```

Aws::DynamoDB::Model::KeyType::HASH);
request.AddKeySchema(yearKeySchema);

Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
    Aws::DynamoDB::Model::KeyType::RANGE);
request.AddKeySchema(yearKeySchema);

Aws::DynamoDB::Model::ProvisionedThroughput throughput;
throughput.WithReadCapacityUnits(
    PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
    PROVISIONED_THROUGHPUT_UNITS);
request.SetProvisionedThroughput(throughput);
request.SetTableName(MOVIE_TABLE_NAME);

std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
const Aws::DynamoDB::Model::CreateTableOutcome &result =
dynamoClient.CreateTable(
    request);
if (!result.IsSuccess()) {
    if (result.GetError().GetErrorCode() ==
        Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
        std::cout << "Table already exists." << std::endl;
        movieTableAlreadyExisted = true;
    }
    else {
        std::cerr << "Failed to create table: "
            << result.GetError().GetMessage();
        return false;
    }
}
// Wait for table to become active.
if (!movieTableAlreadyExisted) {
    std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
        << "' to become active..." << std::endl;
    if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME, clientConfiguration))
    {
        return false;
    }
    std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
        << std::endl;
}
return true;
}

/// Delete a DynamoDB table.
/*!
\sa deleteDynamoTable()
\param tableName: The DynamoDB table's name.
\param clientConfiguration: Aws client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteDynamoTable(const Aws::String &tableName,
                                            const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result = dynamoClient.DeleteTable(
        request);
    if (result.IsSuccess()) {

```

```
        std::cout << "Your table \""
                << result.GetResult().GetTableDescription().GetTableName()
                << " was deleted.\n";
    }
else {
    std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                << std::endl;
}

return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
\sa waitTableActive()
\param waitTableActive: The DynamoDB table's name.
\param clientConfiguration: Aws client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                         const Aws::Client::ClientConfiguration
                                         &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
            request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                    << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for C++ API Reference*.

## Query a table using PartiQL

The following code example shows how to:

- Get an item by running a SELECT statement.
- Add an item by running an INSERT statement.

- Update an item by running an UPDATE statement.
- Delete an item by running a DELETE statement.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// 1. Create a table. (CreateTable)
if (AwsDoc::DynamoDB::createDynamoDBTable(AwsDoc::DynamoDB::MOVIE_TABLE_NAME,
    clientConfig)) {

    AwsDoc::DynamoDB::partiqlExecuteScenario(clientConfig);

    // 7. Delete the table. (DeleteTable)
    AwsDoc::DynamoDB::deleteDynamoTable(AwsDoc::DynamoDB::MOVIE_TABLE_NAME,
        clientConfig);
}

//! Scenario to modify and query a DynamoDB table using single PartiQL statements.
/*
\sa partiqlExecuteScenario()
\param clientConfiguration: Aws client configuration.
\return bool: Function succeeded.
*/
bool
AwsDoc::DynamoDB::partiqlExecuteScenario(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
            "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it? ",
            1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {"'
            << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"'
            << INFO_KEY << "': ?}";

        request.SetStatement(sqlStream.str());

        // Create the parameter attributes.
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
```

```

        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
    ALLOCATION_TAG.c_str());
plotAttribute->SetS(plot);
infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
attributes.push_back(infoMapAttribute);
request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add a movie: " << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    }

    std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
        << std::endl;

// 3. Get the data for the movie using a "Select" statement. (ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM \\" << MOVIE_TABLE_NAME << "\\ WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
        request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to retrieve movie information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    else {
        // Print the retrieved movie information.
        const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

        if (items.size() == 1) {
            printMovieInfo(items[0]);
        }
        else {
            std::cerr << "Error: " << items.size() << " movies were retrieved. "
                << " There should be only one movie." << std::endl;
        }
    }
}

```

```

// 4. Update the data for the movie using an "Update" statement.
(ExecuteStatement)
{
    rating = askQuestionForFloatRange(
        Aws::String("\nLet's update your movie.\nYou rated it  ") +
        std::to_string(rating)
        + ", what new rating would you give it? ", 1, 10);

    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "UPDATE \"\" << MOVIE_TABLE_NAME << "\" SET "
        << INFO_KEY << "." << RATING_KEY << "=? WHERE "
        << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to update a movie: "
            << outcome.GetError().GetMessage();
        return false;
    }
}

std::cout << "\nUpdated " << title << " with new attributes:" << std::endl;

// 5. Get the updated data for the movie using a "Select" statement.
(ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM \"\" << MOVIE_TABLE_NAME << "\" WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to retrieve the movie information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    else {
        const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();
    }
}

```

```

        if (items.size() == 1) {
            printMovieInfo(items[0]);
        }
        else {
            std::cerr << "Error: " << items.size() << " movies were retrieved. "
                << " There should be only one movie." << std::endl;
        }
    }

    std::cout << "Deleting the movie" << std::endl;

    // 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "DELETE FROM `"
            << MOVIE_TABLE_NAME << "` WHERE "
            << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
        dynamoClient.ExecuteStatement(
            request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movie: "
                << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    std::cout << "Movie successfully deleted." << std::endl;
    return true;
}

//! Create a DynamoDB table.
/*!
 \sa createDynamoDBTable()
 \param tableName: The DynamoDB table's name.
 \param clientConfiguration: Aws client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::createDynamoDBTable(const Aws::String &tableName,
                                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
            Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        titleAttributeDefinition.SetAttributeName(TITLE_KEY);
        titleAttributeDefinition.SetAttributeType(

```

```

Aws::DynamoDB::Model::ScalarAttributeType::S);
request.AddAttributeDefinitions(yearAttributeDefinition);

Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
    Aws::DynamoDB::Model::KeyType::HASH);
request.AddKeySchema(yearKeySchema);

Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
    Aws::DynamoDB::Model::KeyType::RANGE);
request.AddKeySchema(titleKeySchema);

Aws::DynamoDB::Model::ProvisionedThroughput throughput;
throughput.WithReadCapacityUnits(
    PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
    PROVISIONED_THROUGHPUT_UNITS);
request.SetProvisionedThroughput(throughput);
request.SetTableName(MOVIE_TABLE_NAME);

std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
const Aws::DynamoDB::Model::CreateTableOutcome &result =
dynamoClient.CreateTable(
    request);
if (!result.IsSuccess()) {
    if (result.GetError().GetErrorCode() ==
        Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
        std::cout << "Table already exists." << std::endl;
        movieTableAlreadyExisted = true;
    }
    else {
        std::cerr << "Failed to create table: "
            << result.GetError().GetMessage();
        return false;
    }
}
// Wait for table to become active.
if (!movieTableAlreadyExisted) {
    std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
        << "' to become active...." << std::endl;
    if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME, clientConfiguration))
    {
        return false;
    }
    std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
        << std::endl;
}
return true;
}

//! Delete a DynamoDB table.
/*!
\sa deleteDynamoTable()
\param tableName: The DynamoDB table's name.
\param clientConfiguration: Aws client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteDynamoTable(const Aws::String &tableName,
                                         const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;

```

```
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result = dynamoClient.DeleteTable(
        request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
            << result.GetResult().GetTableDescription().GetTableName()
            << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
            << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
 \sa waitTableActive()
 \param waitTableActive: The DynamoDB table's name.
 \param clientConfiguration: Aws client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                         const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
            request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for C++ API Reference*.

## EventBridge examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon EventBridge.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2655\)](#)

## Actions

### Add a Lambda function target

The following code example shows how to add an AWS Lambda function target to an Amazon EventBridge event.

#### SDK for C++

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutTargetsRequest.h>
#include <aws/events/model/PutTargetsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Add the target.

```
Aws::CloudWatchEvents::EventBridgeClient cwe;

Aws::CloudWatchEvents::Model::Target target;
target.SetArn(lambda_arn);
target.SetId(target_id);

Aws::CloudWatchEvents::Model::PutTargetsRequest request;
request.SetRule(rule_name);
request.AddTargets(target);

auto putTargetsOutcome = cwe.PutTargets(request);
if (!putTargetsOutcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch events target for rule "
    << rule_name << ":" <<
    putTargetsOutcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout <<
    "Successfully created CloudWatch events target for rule "
    << rule_name << std::endl;
```

```
}
```

- For API details, see [PutTargets](#) in *AWS SDK for C++ API Reference*.

## Create a scheduled rule

The following code example shows how to create an Amazon EventBridge scheduled rule.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutRuleRequest.h>
#include <aws/events/model/PutRuleResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Create the rule.

```
Aws::CloudWatchEvents::EventBridgeClient cwe;
Aws::CloudWatchEvents::Model::PutRuleRequest request;
request.SetName(rule_name);
request.SetRoleArn(role_arn);
request.SetScheduleExpression("rate(5 minutes)");
request.SetState(Aws::CloudWatchEvents::Model::RuleState::ENABLED);

auto outcome = cwe.PutRule(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch events rule " <<
        rule_name << ":" << outcome.GetError().GetMessage() <<
        std::endl;
}
else
{
    std::cout << "Successfully created CloudWatch events rule " <<
        rule_name << " with resulting Arn " <<
        outcome.GetResult().GetRuleArn() << std::endl;
}
```

- For API details, see [PutRule](#) in *AWS SDK for C++ API Reference*.

## Send events

The following code example shows how to send Amazon EventBridge events.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Include the required files.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutEventsRequest.h>
#include <aws/events/model/PutEventsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Send the event.

```
Aws::CloudWatchEvents::EventBridgeClient cwe;

Aws::CloudWatchEvents::Model::PutEventsRequestEntry event_entry;
event_entry.SetDetail(MakeDetails(event_key, event_value));
event_entry.SetDetailType("sampleSubmitted");
event_entry.AddResources(resource_arn);
event_entry.SetSource("aws-sdk-cpp-cloudwatch-example");

Aws::CloudWatchEvents::Model::PutEventsRequest request;
request.AddEntries(event_entry);

auto outcome = cwe.PutEvents(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to post CloudWatch event: " <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully posted CloudWatch event" << std::endl;
}
```

- For API details, see [PutEvents](#) in *AWS SDK for C++ API Reference*.

## IAM examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with AWS Identity and Access Management.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2657\)](#)
- [Scenarios \(p. 2675\)](#)

## Actions

### Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::attachRolePolicy(const Aws::String &roleName,
                                    const Aws::String &policyArn,
                                    const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::ListAttachedRolePoliciesRequest list_request;
    list_request.SetRoleName(roleName);

    bool done = false;
    while (!done) {
        auto list_outcome = iam.ListAttachedRolePolicies(list_request);
        if (!list_outcome.IsSuccess()) {
            std::cerr << "Failed to list attached policies of role " <<
                roleName << ":" << list_outcome.GetError().GetMessage() <<
                std::endl;
            return false;
        }

        const auto &policies = list_outcome.GetResult().GetAttachedPolicies();
        if (std::any_of(policies.cbegin(), policies.cend(),
                      [=](const Aws::IAM::Model::AttachedPolicy &policy) {
                          [=](const Aws::IAM::Model::AttachedPolicy &policy) {
                              return policy.GetPolicyArn() == policyArn;
                          })) {
            std::cout << "Policy " << policyArn <<
                " is already attached to role " << roleName << std::endl;
            return true;
        }

        done = !list_outcome.GetResult().GetIsTruncated();
        list_request.SetMarker(list_outcome.GetResult().GetMarker());
    }

    Aws::IAM::Model::AttachRolePolicyRequest request;
    request.SetRoleName(roleName);
    request.SetPolicyArn(policyArn);

    Aws::IAM::Model::AttachRolePolicyOutcome outcome = iam.AttachRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to attach policy " << policyArn << " to role " <<
            roleName << ":" << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully attached policy " << policyArn << " to role " <<
            roleName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [AttachRolePolicy](#) in [AWS SDK for C++ API Reference](#).

## Attach an inline policy to a role

The following code example shows how to attach an inline policy to an IAM role.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::putRolePolicy(
    const Aws::String &roleName,
    const Aws::String &policyName,
    const Aws::String &policyDocument,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iamClient(clientConfig);
    Aws::IAM::Model::PutRolePolicyRequest request;

    request.SetRoleName(roleName);
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(policyDocument);

    Aws::IAM::Model::PutRolePolicyOutcome outcome = iamClient.PutRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error putting policy on role. " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully put the role policy." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [PutRolePolicy](#) in *AWS SDK for C++ API Reference*.

## Create a policy

The following code example shows how to create an IAM policy.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::String AwsDoc::IAM::createPolicy(const Aws::String &policyName,
                                         const Aws::String &rsrArn,
                                         const Aws::Client::ClientConfiguration
                                         &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreatePolicyRequest request;
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(BuildSamplePolicyDocument(rsrArn));

    Aws::IAM::Model::CreatePolicyOutcome outcome = iam.CreatePolicy(request);
    Aws::String result;
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy " << policyName << ":" <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        result = outcome.GetResult().GetPolicy().GetArn();
        std::cout << "Successfully created policy " << policyName <<
```

```
        std::endl;
    }

    return result;
}

Aws::String AwsDoc::IAM::BuildSamplePolicyDocument(const Aws::String &rsrc_arn) {
    std::stringstream stringStream;
    stringStream << "{"
        << "    \"Version\": \"2012-10-17\","
        << "    \"Statement\": ["
        << "        {"
            << "            \"Effect\": \"Allow\","
            << "            \"Action\": \"logs:CreateLogGroup\","
            << "            \"Resource\": \""
            << rsrc_arn
            << "\"
            << "        },"
            << "        {"
                << "            \"Effect\": \"Allow\","
                << "            \"Action\": ["
                    << "                \"dynamodb>DeleteItem\","
                    << "                \"dynamodb>GetItem\","
                    << "                \"dynamodb>PutItem\","
                    << "                \"dynamodb>Scan\","
                    << "                \"dynamodb>UpdateItem\""
                << "            ],"
                << "            \"Resource\": \""
                << rsrc_arn
                << "\"
                << "        }"
            << "    ]"
        << "}";
    }

    return stringStream.str();
}
```

- For API details, see [CreatePolicy](#) in *AWS SDK for C++ API Reference*.

## Create a role

The following code example shows how to create an IAM role.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::createIamRole(
    const Aws::String &roleName,
    const Aws::String &policy,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::CreateRoleRequest request;

    request.SetRoleName(roleName);
    request.SetAssumeRolePolicyDocument(policy);

    Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating role. " <<
```

```
        outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const Aws::IAM::Model::Role iamRole = outcome.GetResult().GetRole();
        std::cout << "Created role " << iamRole.GetRoleName() << "\n";
        std::cout << "ID: " << iamRole.GetRoleId() << "\n";
        std::cout << "ARN: " << iamRole.GetArn() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [CreateRole](#) in *AWS SDK for C++ API Reference*.

## Create a user

The following code example shows how to create an IAM user.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::CreateUserRequest create_request;
create_request.SetUserName(userName);

auto create_outcome = iam.CreateUser(create_request);
if (!create_outcome.IsSuccess()) {
    std::cerr << "Error creating IAM user " << userName << ":" <<
    create_outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully created IAM user " << userName << std::endl;
}

return create_outcome.IsSuccess();
```

- For API details, see [CreateUser](#) in *AWS SDK for C++ API Reference*.

## Create an access key

The following code example shows how to create an IAM access key.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::String AwsDoc::IAM::createAccessKey(const Aws::String &userName,
                                         const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateAccessKeyRequest request;
```

```
    request.SetUserName(userName);

    Aws::String result;
    Aws::IAM::Model::CreateAccessKeyOutcome outcome = iam.CreateAccessKey(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating access key for IAM user " << userName
             << ":" << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &accessKey = outcome.GetResult().GetAccessKey();
        std::cout << "Successfully created access key for IAM user " <<
            userName << std::endl << " aws_access_key_id = " <<
            accessKey.GetAccessKeyId() << std::endl <<
            " aws_secret_access_key = " << accessKey.GetSecretAccessKey() <<
            std::endl;
        result = accessKey.GetAccessKeyId();
    }

    return result;
}
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for C++ API Reference*.

## Create an alias for an account

The following code example shows how to create an alias for an IAM account.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::createAccountAlias(const Aws::String &aliasName,
                                      const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::CreateAccountAliasRequest request;
    request.SetAccountAlias(aliasName);

    Aws::IAM::Model::CreateAccountAliasOutcome outcome = iam.CreateAccountAlias(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating account alias " << aliasName << ":" "
             << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created account alias " << aliasName <<
            std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [CreateAccountAlias](#) in *AWS SDK for C++ API Reference*.

## Delete a policy

The following code example shows how to delete an IAM policy.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::deletePolicy(const Aws::String &policyArn,
                                const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeletePolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.DeletePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting policy with arn " << policyArn << ":" 
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted policy with arn " << policyArn
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeletePolicy](#) in *AWS SDK for C++ API Reference*.

## Delete a server certificate

The following code example shows how to delete an IAM server certificate.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::deleteServerCertificate(const Aws::String &certificateName,
                                           const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeleteServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    const auto outcome = iam.DeleteServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorCode() != Aws::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error deleting server certificate " << certificateName <<
                      ":" << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
    else {
        std::cout << "Successfully deleted server certificate " << certificateName
                  << std::endl;
    }
}
```

```
        }
    return result;
}
```

- For API details, see [DeleteServerCertificate](#) in *AWS SDK for C++ API Reference*.

## Delete a user

The following code example shows how to delete an IAM user.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::DeleteUserRequest request;
request.SetUserName(userName);
auto outcome = iam.DeleteUser(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error deleting IAM user " << userName << ":" <<
        outcome.GetError().GetMessage() << std::endl;;
}
else {
    std::cout << "Successfully deleted IAM user " << userName << std::endl;
}

return outcome.IsSuccess();
```

- For API details, see [DeleteUser](#) in *AWS SDK for C++ API Reference*.

## Delete an access key

The following code example shows how to delete an IAM access key.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::deleteAccessKey(const Aws::String &userName,
                                    const Aws::String &accessKeyId,
                                    const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyId);

    auto outcome = iam.DeleteAccessKey(request);

    if (!outcome.IsSuccess()) {
```

```
        std::cerr << "Error deleting access key " << accessKeyID << " from user "
        << userName << ": " << outcome.GetError().GetMessage() <<
        std::endl;
    }
else {
    std::cout << "Successfully deleted access key " << accessKeyID
    << " for IAM user " << userName << std::endl;
}

return outcome.IsSuccess();
}
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for C++ API Reference*.

## Delete an account alias

The following code example shows how to delete an IAM account alias.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::deleteAccountAlias(const Aws::String &accountAlias,
                                      const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccountAliasRequest request;
    request.SetAccountAlias(accountAlias);

    const auto outcome = iam.DeleteAccountAlias(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting account alias " << accountAlias << ": "
        << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted account alias " << accountAlias <<
        std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteAccountAlias](#) in *AWS SDK for C++ API Reference*.

## Detach a policy from a role

The following code example shows how to detach an IAM policy from a role.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::IAM::IAMClient iam(clientConfig);
```

```
Aws::IAM::Model::DetachRolePolicyRequest detachRequest;
detachRequest.SetRoleName(roleName);
detachRequest.SetPolicyArn(policyArn);

auto detachOutcome = iam.DetachRolePolicy(detachRequest);
if (!detachOutcome.IsSuccess()) {
    std::cerr << "Failed to detach policy " << policyArn << " from role "
    << roleName << ": " << detachOutcome.GetError().GetMessage() <<
    std::endl;
}
else {
    std::cout << "Successfully detached policy " << policyArn << " from role "
    << roleName << std::endl;
}

return detachOutcome.IsSuccess();
```

- For API details, see [DetachRolePolicy](#) in *AWS SDK for C++ API Reference*.

## Get a policy

The following code example shows how to get an IAM policy.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::getPolicy(const Aws::String &policyArn,
                             const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetPolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.GetPolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error getting policy " << policyArn << ":" <<
        outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &policy = outcome.GetResult().GetPolicy();
        std::cout << "Name: " << policy.GetPolicyName() << std::endl <<
        "ID: " << policy.GetPolicyId() << std::endl << "Arn: " <<
        policy.GetArn() << std::endl << "Description: " <<
        policy.GetDescription() << std::endl << "CreateDate: " <<
        policy.GetCreateDate().ToGmtString(Aws::Utils::DateFormat::ISO_8601)
        << std::endl;
    }
    return outcome.IsSuccess();
}
```

- For API details, see [GetPolicy](#) in *AWS SDK for C++ API Reference*.

## Get a server certificate

The following code example shows how to get an IAM server certificate.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::getServerCertificate(const Aws::String &certificateName,
                                         const Aws::Client::ClientConfiguration
                                         &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    auto outcome = iam.GetServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorCode() != Aws::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error getting server certificate " << certificateName <<
                ":" << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                  << "' not found." << std::endl;
        }
    }
    else {
        const auto &certificate = outcome.GetResult().GetServerCertificate();
        std::cout << "Name: " <<
            certificate.GetServerCertificateMetadata().GetServerCertificateName()
            << std::endl << "Body: " << certificate.GetCertificateBody() <<
            std::endl << "Chain: " << certificate.GetCertificateChain() <<
            std::endl;
    }
}

return result;
}
```

- For API details, see [GetServerCertificate](#) in *AWS SDK for C++ API Reference*.

## Get data about the last use of an access key

The following code example shows how to get data about the last use of an IAM access key.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::accessKeyLastUsed(const Aws::String &secretKeyId,
                                      const Aws::Client::ClientConfiguration
                                      &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetAccessKeyLastUsedRequest request;

    request.SetAccessKeyId(secretKeyId);

    Aws::IAM::Model::GetAccessKeyLastUsedOutcome outcome = iam.GetAccessKeyLastUsed(
```

```
        request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error querying last used time for access key " <<
            secretKeyID << ":" << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        Aws::String lastUsedTimeString =
            outcome.GetResult()
                .GetAccessKeyLastUsed()
                .GetLastUsedDate()
                .ToGmtString(Aws::Utils::DateFormat::ISO_8601);
        std::cout << "Access key " << secretKeyID << " last used at time " <<
            lastUsedTimeString << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [GetAccessKeyLastUsed](#) in *AWS SDK for C++ API Reference*.

## List a user's access keys

The following code example shows how to list a user's IAM access keys.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::listAccessKeys(const Aws::String &userName,
                                  const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccessKeysRequest request;
    request.SetUserName(userName);

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccessKeys(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list access keys for user " << userName
                  << ":" << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "UserName" <<
                std::setw(30) << "KeyID" << std::setw(20) << "Status" <<
                std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &keys = outcome.GetResult().GetAccessKeyMetadata();
        const Aws::String DATE_FORMAT = "%Y-%m-%d";

        for (const auto &key: keys) {
            Aws::String statusString =
                Aws::IAM::Model::StatusTypeMapper::GetNameForStatusType(

```

```
        key.GetStatus());
    std::cout << std::left << std::setw(32) << key.GetUserName() <<
    std::setw(30) << key.GetAccessKeyId() << std::setw(20) <<
    statusString << std::setw(20) <<
    key.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
    std::endl;
}

if (outcome.GetResult().GetIsTruncated()) {
    request.SetMarker(outcome.GetResult().GetMarker());
}
else {
    done = true;
}
}

return true;
}
```

- For API details, see [ListAccessKeys](#) in *AWS SDK for C++ API Reference*.

## List account aliases

The following code example shows how to list IAM account aliases.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool
AwsDoc::IAM::listAccountAliases(const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccountAliasesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccountAliases(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list account aliases: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        const auto &aliases = outcome.GetResult().GetAccountAliases();
        if (!header) {
            if (aliases.size() == 0) {
                std::cout << "Account has no aliases" << std::endl;
                break;
            }
            std::cout << std::left << std::setw(32) << "Alias" << std::endl;
            header = true;
        }

        for (const auto &alias: aliases) {
            std::cout << std::left << std::setw(32) << alias << std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
```

```
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else {
        done = true;
    }
}

return true;
}
```

- For API details, see [ListAccountAliases](#) in *AWS SDK for C++ API Reference*.

## List policies

The following code example shows how to list IAM policies.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::listPolicies(const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT("%Y-%m-%d");
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListPoliciesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListPolicies(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam policies: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                std::setw(64) << "Description" << std::setw(12) <<
                "CreateDate" << std::endl;
            header = true;
        }

        const auto &policies = outcome.GetResult().GetPolicies();
        for (const auto &policy: policies) {
            std::cout << std::left << std::setw(55) <<
                policy.GetPolicyName() << std::setw(30) <<
                policy.GetPolicyId() << std::setw(80) << policy.GetArn() <<
                std::setw(64) << policy.GetDescription() << std::setw(12) <<
                policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
                std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }
}
```

```
        }

    return true;
}
```

- For API details, see [ListPolicies](#) in *AWS SDK for C++ API Reference*.

## List server certificates

The following code example shows how to list IAM server certificates.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::listServerCertificates(
    const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";

    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListServerCertificatesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListServerCertificates(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list server certificates: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                std::setw(14) << "UploadDate" << std::setw(14) <<
                "ExpirationDate" << std::endl;
            header = true;
        }

        const auto &certificates =
            outcome.GetResult().GetServerCertificateMetadataList();

        for (const auto &certificate: certificates) {
            std::cout << std::left << std::setw(55) <<
                certificate.GetServerCertificateName() << std::setw(30) <<
                certificate.GetServerCertificateId() << std::setw(80) <<
                certificate.GetArn() << std::setw(14) <<
                certificate.GetUploadDate().ToGmtString(DATE_FORMAT.c_str()) <<
                std::setw(14) <<
                certificate.GetExpiration().ToGmtString(DATE_FORMAT.c_str()) <<
                std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }
}
```

```
        }
    }

    return true;
}
```

- For API details, see [ListServerCertificates](#) in *AWS SDK for C++ API Reference*.

## List users

The following code example shows how to list IAM users.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::listUsers(const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListUsersRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListUsers(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam users:" <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "Name" <<
                std::setw(30) << "ID" << std::setw(64) << "Arn" <<
                std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &users = outcome.GetResult().GetUsers();
        for (const auto &user: users) {
            std::cout << std::left << std::setw(32) << user.GetUserName() <<
                std::setw(30) << user.GetUserId() << std::setw(64) <<
                user.GetArn() << std::setw(20) <<
                user.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
                << std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- For API details, see [ListUsers](#) in *AWS SDK for C++ API Reference*.

## Update a server certificate

The following code example shows how to update an IAM server certificate.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::updateServerCertificate(const Aws::String &currentCertificateName,
                                           const Aws::String &newCertificateName,
                                           const Aws::Client::ClientConfiguration
                                           &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateServerCertificateRequest request;
    request.SetServerCertificateName(currentCertificateName);
    request.SetNewServerCertificateName(newCertificateName);

    auto outcome = iam.UpdateServerCertificate(request);
    bool result = true;
    if (outcome.IsSuccess()) {
        std::cout << "Server certificate " << currentCertificateName
              << " successfully renamed as " << newCertificateName
              << std::endl;
    }
    else {
        if (outcome.GetError().GetErrorCode() != Aws::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error changing name of server certificate " <<
                currentCertificateName << " to " << newCertificateName << ":" <<
                outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << currentCertificateName
                  << "' not found." << std::endl;
        }
    }
    return result;
}
```

- For API details, see [UpdateServerCertificate](#) in *AWS SDK for C++ API Reference*.

## Update a user

The following code example shows how to update an IAM user.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::updateUser(const Aws::String &currentUserNames,
```

```
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::UpdateUserRequest request;
    request.SetUserName(currentUserName);
    request.SetNewUserName(newUserName);

    auto outcome = iam.UpdateUser(request);
    if (outcome.IsSuccess()) {
        std::cout << "IAM user " << currentUserName <<
                    " successfully updated with new user name " << newUserName <<
                    std::endl;
    }
    else {
        std::cerr << "Error updating user name for IAM user " << currentUserName <<
                    ":" << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [UpdateUser](#) in *AWS SDK for C++ API Reference*.

## Update an access key

The following code example shows how to update an IAM access key.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::IAM::updateAccessKey(const Aws::String &userName,
                                    const Aws::String &accessKeyID,
                                    Aws::IAM::Model::StatusType status,
                                    const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);
    request.SetStatus(status);

    auto outcome = iam.UpdateAccessKey(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated status of access key "
                    << accessKeyID << " for user " << userName << std::endl;
    }
    else {
        std::cerr << "Error updated status of access key " << accessKeyID <<
                    " for user " << userName << ":" <<
                    outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [UpdateAccessKey](#) in *AWS SDK for C++ API Reference*.

## Scenarios

### Create a user and assume a role

The following code example shows how to:

- Create a user who has no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list Amazon S3 buckets using temporary credentials.
- Delete the policy, role, and user.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace AwsDoc {
    namespace IAM {
        //! Cleanup by deleting created entities.
        /*!
         \sa DeleteCreatedEntities
         \param client: IAM client.
         \param role: IAM role.
         \param user: IAM user.
         \param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                         const Aws::IAM::Model::Role &role,
                                         const Aws::IAM::Model::User &user,
                                         const Aws::IAM::Model::Policy &policy);
    }
}

//! Scenario to create an IAM user, create an IAM role, and apply the role to the user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
// necessary to
//     create a custom policy).
/*!
 \sa iamCreateUserAssumeRoleScenario
 \param clientConfig: Aws client configuration.
 \return bool: Successful completion.
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
    const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
            Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);
```

```

Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
if (!outcome.IsSuccess()) {
    std::cout << "Error creating IAM user " << userName << ":" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}
else {
    std::cout << "Successfully created IAM user " << userName << std::endl;
}

user = outcome.GetResult().GetUser();
}

// 2. Create a role.
{
    // Get the IAM user for the current client in order to access its ARN.
    Aws::String iamUserArn;
    {
        Aws::IAM::Model:: GetUserRequest request;
        Aws::IAM::Model:: GetUserOutcome outcome = client.GetUser(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error getting Iam user. " <<
                outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully retrieved Iam user "
                << outcome.GetResult().GetUser().GetUserName()
                << std::endl;
        }
    }

    iamUserArn = outcome.GetResult().GetUser().GetArn();
}

Aws::IAM::Model::CreateRoleRequest request;

Aws::String uuid = Aws::Utils::UUID::RandomUUID();
Aws::String roleName = "iam-demo-role-" +
    Aws::Utils::StringUtils::ToLower(uuid.c_str());
request.SetRoleName(roleName);

// Build policy document for role.
Aws::Utils::Document jsonStatement;
jsonStatement.WithString("Effect", "Allow");

Aws::Utils::Document jsonPrincipal;
jsonPrincipal.WithString("AWS", iamUserArn);
jsonStatement.WithObject("Principal", jsonPrincipal);
jsonStatement.WithString("Action", "sts:AssumeRole");
jsonStatement.WithObject("Condition", Aws::Utils::Document());

Aws::Utils::Document policyDocument;
policyDocument.WithString("Version", "2012-10-17");

Aws::Utils::Array< Aws::Utils::Document> statements(1);
statements[0] = jsonStatement;
policyDocument.WithArray("Statement", statements);

std::cout << "Setting policy for role\n  "
    << policyDocument.View().WriteCompact() << std::endl;

// Set role policy document as JSON string.
request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

```

```

Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error creating role. " <<
        outcome.GetError().GetMessage() << std::endl;

    DeleteCreatedEntities(client, role, user, policy);
    return false;
}
else {
    std::cout << "Successfully created a role with name " << roleName
        << std::endl;
}

role = outcome.GetResult().GetRole();
}

// 3. Create an IAM policy.
{
    Aws::IAM::Model::CreatePolicyRequest request;
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String policyName = "iam-demo-policy-" +
        Aws::Utils::StringUtils::ToLower(uuid.c_str());
    request.SetPolicyName(policyName);

    // Build IAM policy document.
    Aws::Utils::Document jsonStatement;
    jsonStatement.WithString("Effect", "Allow");
    jsonStatement.WithString("Action", "s3>ListAllMyBuckets");
    jsonStatement.WithString("Resource", "arn:aws:s3:::*");

    Aws::Utils::Document policyDocument;
    policyDocument.WithString("Version", "2012-10-17");

    Aws::Utils::Array< Aws::Utils::Document> statements(1);
    statements[0] = jsonStatement;
    policyDocument.WithArray("Statement", statements);

    std::cout << "Creating a policy.\n" << policyDocument.View().WriteCompact()
        << std::endl;

    // Set IAM policy document as JSON string.
    request.SetPolicyDocument(policyDocument.View().WriteCompact());

    Aws::IAM::Model::CreatePolicyOutcome outcome = client.CreatePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy. " <<
            outcome.GetError().GetMessage() << std::endl;

        DeleteCreatedEntities(client, role, user, policy);
        return false;
    }
    else {
        std::cout << "Successfully created a policy with name, " << policyName <<
            "." << std::endl;
    }

    policy = outcome.GetResult().GetPolicy();
}

// 4. Assume the new role using the AWS Security Token Service (STS).
Aws::STS::Model::Credentials credentials;
{
    Aws::STS::STSClient stsClient(clientConfig);

    Aws::STS::Model::AssumeRoleRequest request;
    request.SetRoleArn(role.GetArn());
}

```

```

Aws::String uuid = Aws::Utils::UUID::RandomUUID();
Aws::String roleSessionName = "iam-demo-role-session-" +
    Aws::Utils::StringUtils::ToLower(uuid.c_str());
request.SetRoleSessionName(roleSessionName);

Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

// Repeatedly call AssumeRole, because there is often a delay
// before the role is available to be assumed.
// Repeat at most 20 times when access is denied.
int count = 0;
while (true) {
    assumeRoleOutcome = stsClient.AssumeRole(request);
    if (!assumeRoleOutcome.IsSuccess()) {
        if (count > 20 || 
            assumeRoleOutcome.GetError().GetErrorCode() != 
            Aws::STS::STSErrors::ACCESS_DENIED) {
            std::cerr << "Error assuming role after 20 tries. " <<
                assumeRoleOutcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        std::this_thread::sleep_for(std::chrono::seconds(1));
    }
    else {
        std::cout << "Successfully assumed the role after " << count
            << " seconds." << std::endl;
        break;
    }
    count++;
}

credentials = assumeRoleOutcome.GetResult().GetCredentials();
}

// 5. List objects in the bucket (This should fail).
{
    Aws::S3::S3Client s3Client(
        Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
            credentials.GetSecretAccessKey(),
            credentials.GetSessionToken()),
        clientConfig);
    Aws::S3::Model::ListBucketsOutcome listBucketsOutcome = s3Client.ListBuckets();
    if (!listBucketsOutcome.IsSuccess()) {
        if (listBucketsOutcome.GetError().GetErrorCode() != 
            Aws::S3::S3Errors::ACCESS_DENIED) {
            std::cerr << "Could not lists buckets. " <<
                listBucketsOutcome.GetError().GetMessage() << std::endl;
        }
        else {
            std::cout
                << "Access to list buckets denied because privileges have not
been applied."
                << std::endl;
        }
    }
    else {
        std::cerr
            << "Successfully retrieved bucket lists when this should not
happen."
            << std::endl;
    }
}

```

```

// 6. Attach the policy to the role.
{
    Aws::IAM::Model::AttachRolePolicyRequest request;
    request.SetRoleName(role.GetRoleName());
    request.WithPolicyArn(policy.GetArn());

    Aws::IAM::Model::AttachRolePolicyOutcome outcome = client.AttachRolePolicy(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy. " <<
            outcome.GetError().GetMessage() << std::endl;

        DeleteCreatedEntities(client, role, user, policy);
        return false;
    }
    else {
        std::cout << "Successfully attached the policy with name, "
            << policy.GetPolicyName() <<
            ", to the role, " << role.GetRoleName() << "." << std::endl;
    }
}

int count = 0;
// 7. List objects in the bucket (this should succeed).
// Repeatedly call ListBuckets, because there is often a delay
// before the policy with ListBucket permissions has been applied to the role.
// Repeat at most 20 times when access is denied.
while (true) {
    Aws::S3::S3Client s3Client(
        Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
            credentials.GetSecretAccessKey(),
            credentials.GetSessionToken()),
        clientConfig);
    Aws::S3::Model::ListBucketsOutcome listBucketsOutcome = s3Client.ListBuckets();
    if (!listBucketsOutcome.IsSuccess()) {
        if ((count > 20) ||
            listBucketsOutcome.GetError().GetErrorType() !=
            Aws::S3::S3Errors::ACCESS_DENIED) {
            std::cerr << "Could not lists buckets after 20 seconds. " <<
                listBucketsOutcome.GetError().GetMessage() << std::endl;
            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        std::this_thread::sleep_for(std::chrono::seconds(1));
    }
    else {
        std::cout << "Successfully retrieved bucket lists after " << count
            << " seconds." << std::endl;
        break;
    }
    count++;
}

// 8. Delete all the created resources.
return DeleteCreatedEntities(client, role, user, policy);
}

bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                         const Aws::IAM::Model::Role &role,
                                         const Aws::IAM::Model::User &user,
                                         const Aws::IAM::Model::Policy &policy) {
    bool result = true;
    if (policy.ArnsHasBeenSet()) {
        // Detach the policy from the role.

```

```
{  
    Aws::IAM::Model::DetachRolePolicyRequest request;  
    request.SetPolicyArn(policy.GetArn());  
    request.SetRoleName(role.GetRoleName());  
  
    Aws::IAM::Model::DetachRolePolicyOutcome outcome = client.DetachRolePolicy(  
        request);  
    if (!outcome.IsSuccess()) {  
        std::cerr << "Error Detaching policy from roles. " <<  
            outcome.GetError().GetMessage() << std::endl;  
        result = false;  
    }  
    else {  
        std::cout << "Successfully detached the policy with arn "  
            << policy.GetArn()  
            << " from role " << role.GetRoleName() << "." << std::endl;  
    }  
}  
  
// Delete the policy.  
{  
    Aws::IAM::Model::DeletePolicyRequest request;  
    request.WithPolicyArn(policy.GetArn());  
  
    Aws::IAM::Model::DeletePolicyOutcome outcome =  
client.DeletePolicy(request);  
    if (!outcome.IsSuccess()) {  
        std::cerr << "Error deleting policy. " <<  
            outcome.GetError().GetMessage() << std::endl;  
        result = false;  
    }  
    else {  
        std::cout << "Successfully deleted the policy with arn "  
            << policy.GetArn() << std::endl;  
    }  
}  
  
}  
  
if (role.RoleIdHasBeenSet()) {  
    // Delete the role.  
    Aws::IAM::Model::DeleteRoleRequest request;  
    request.SetRoleName(role.GetRoleName());  
  
    Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);  
    if (!outcome.IsSuccess()) {  
        std::cerr << "Error deleting role. " <<  
            outcome.GetError().GetMessage() << std::endl;  
        result = false;  
    }  
    else {  
        std::cout << "Successfully deleted the role with name "  
            << role.GetRoleName() << std::endl;  
    }  
}  
  
if (user.ArnHasBeenSet()) {  
    // Delete the user.  
    Aws::IAM::Model::DeleteUserRequest request;  
    request.WithUserName(user.GetUserName());  
  
    Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);  
    if (!outcome.IsSuccess()) {  
        std::cerr << "Error deleting user. " <<  
            outcome.GetError().GetMessage() << std::endl;  
        result = false;  
    }  
}
```

```
        }
    else {
        std::cout << "Successfully deleted the user with name "
             << user.GetUserName() << std::endl;
    }
}

return result;
}
```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## Amazon S3 examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon Simple Storage Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2681\)](#)
- [Scenarios \(p. 2693\)](#)

## Actions

### Add a policy to a bucket

The following code example shows how to add a policy to an S3 bucket.

#### SDK for C++

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::PutBucketPolicy(const Aws::String &bucketName,
                                 const Aws::String &policyBody,
                                 const Aws::Client::ClientConfiguration &clientConfig)
{
```

```

Aws::S3::S3Client s3_client(clientConfig);

std::shared_ptr<Aws::StringStream> request_body =
    Aws::MakeShared<Aws::StringStream>("");
*request_body << policyBody;

Aws::S3::Model::PutBucketPolicyRequest request;
request.SetBucket(bucketName);
request.SetBody(request_body);

Aws::S3::Model::PutBucketPolicyOutcome outcome =
    s3_client.PutBucketPolicy(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: PutBucketPolicy: "
        << outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Set the following policy body for the bucket '" <<
        bucketName << ":" << std::endl << std::endl;
    std::cout << policyBody << std::endl;
}

return outcome.IsSuccess();
}

///! Build a policy JSON string.
/*!
 \sa GetPolicyString()
 \param userArn Aws user Amazon Resource Name (ARN).
     For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\_identifiers.html#identifiers-arns.
 \param bucketName Name of a bucket.
 */
Aws::String GetPolicyString(const Aws::String &userArn,
                           const Aws::String &bucketName) {
    return
        "{\n"
        "  \"Version\": \"2012-10-17\",\\n"
        "  \"Statement\":[\n"
        "    {\n"
        "      \"Sid\": \"1\",\\n"
        "      \"Effect\": \"Allow\",\\n"
        "      \"Principal\": {\\n"
        "        \"AWS\": \""
        + userArn +
        "\",\\n\"},\\n"
        "      \"Action\": [ \"s3:GetObject\" ],\\n"
        "      \"Resource\": [ \"arn:aws:s3:::"
        + bucketName +
        "/*\" ]\\n"
        "    }\\n"
        "  ]\\n"
        "}";
}

```

- For API details, see [PutBucketPolicy in AWS SDK for C++ API Reference](#).

### Copy an object from one bucket to another

The following code example shows how to copy an S3 object from one bucket to another.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::CopyObject(const Aws::String &objectKey, const Aws::String
    &fromBucket, const Aws::String &toBucket,
                                const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
        .WithKey(objectKey)
        .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: CopyObject: " <<
            err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully copied " << objectKey << " from " << fromBucket <<
            " to " << toBucket << "." << std::endl;
    }
    return outcome.IsSuccess();
}
```

- For API details, see [CopyObject](#) in *AWS SDK for C++ API Reference*.

## Create a bucket

The following code example shows how to create an S3 bucket.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::CreateBucket(const Aws::String &bucketName,
                               const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    //TODO(user): Change the bucket location constraint enum to your target Region.
    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(
            Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
```

```
        auto err = outcome.GetError();
        std::cerr << "Error: CreateBucket: " <<
            err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Created bucket " << bucketName <<
            " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for C++ API Reference*.

## Delete a policy from a bucket

The following code example shows how to delete a policy from an S3 bucket.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::DeleteBucketPolicy(const Aws::String &bucketName,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
    client.DeleteBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: DeleteBucketPolicy: " <<
            err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteBucketPolicy](#) in *AWS SDK for C++ API Reference*.

## Delete an empty bucket

The following code example shows how to delete an empty S3 bucket.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::DeleteBucket(const Aws::String &bucketName,
                               const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: DeleteBucket: " <<
            err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteBucket](#) in *AWS SDK for C++ API Reference*.

## Delete an object

The following code example shows how to delete an S3 object.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::DeleteObject(const Aws::String &objectKey,
                               const Aws::String &fromBucket,
                               const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
        .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: DeleteObject: " <<
            err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted the object." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteObject](#) in *AWS SDK for C++ API Reference*.

## Delete the website configuration from a bucket

The following code example shows how to delete the website configuration from an S3 bucket.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::DeleteBucketWebsite(const Aws::String &bucketName,
                                      const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
        client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: DeleteBucketWebsite: " <<
            err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [DeleteBucketWebsite](#) in *AWS SDK for C++ API Reference*.

## Get an object from a bucket

The following code example shows how to read data from an object in an S3 bucket.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::GetObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: GetObject: " <<
            err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
    }
}
```

```
        }
    else {
        std::cout << "Successfully retrieved '" << objectKey << "' from ''"
             << fromBucket << '.' << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [GetObject](#) in *AWS SDK for C++ API Reference*.

## Get the ACL of an object

The following code example shows how to get the access control list (ACL) of an S3 object.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::GetBucketAcl(const Aws::String &bucketName,
                               const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
        s3_client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: GetBucketAcl: "
              << err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
    }
    else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            std::cout << "For bucket " << bucketName << ": "
                  << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type: "
                      << GetGranteeTypeString(grantee.GetType()) << std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name: "
                      << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                      << grantee.GetEmailAddress() << std::endl;
            }
        }
    }
}
```

```
        if (grantee.IDHasBeenSet()) {
            std::cout << "ID:           "
                << grantee.GetID() << std::endl;
        }

        if (grantee.URIHasBeenSet()) {
            std::cout << "URI:           "
                << grantee.GetURI() << std::endl;
        }

        std::cout << "Permission:    " <<
            GetPermissionString(grant.GetPermission()) <<
            std::endl << std::endl;
    }

    return outcome.IsSuccess();
}

///! Routine which converts a built-in type enumeration to a human-readable string.
/*!
\sa GetGranteeTypeString()
\param type Type enumeration.
*/
Aws::String GetGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

///! Routine which converts a built-in type enumeration to a human-readable string.
/*!
\sa GetPermissionString()
\param permission Permission enumeration.
*/
Aws::String GetPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                "objects in this bucket, and read/write this "
                "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }
}
```

```
        return "Permission unknown";
    }
```

- For API details, see [GetObjectAcl](#) in *AWS SDK for C++ API Reference*.

## Get the policy for a bucket

The following code example shows how to get the policy for an S3 bucket.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::GetBucketPolicy(const Aws::String &bucketName,
                                  const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
        s3_client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: GetBucketPolicy: "
              << err.ExceptionName() << ":" << err.Message() << std::endl;
    }
    else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Retrieve the policy for bucket '" << bucketName << "' :\n\n" <<
            policy_stream.str() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- For API details, see [GetBucketPolicy](#) in *AWS SDK for C++ API Reference*.

## Get the website configuration for a bucket

The following code example shows how to get the website configuration for an S3 bucket.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::GetWebsiteConfig(const Aws::String &bucketName,
                                    const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
        s3_client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
              << err.GetMessage() << std::endl;
    }
    else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult = outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
              << std::endl << std::endl
              << "For bucket '" << bucketName << "' :"
              << std::endl
              << "Index page :"
              << websiteResult.GetIndexDocument().GetSuffix()
              << std::endl
              << "Error page: "
              << websiteResult.GetErrorDocument().GetKey()
              << std::endl;
    }

    return outcome.IsSuccess();
}
```

- For API details, see [GetBucketWebsite](#) in *AWS SDK for C++ API Reference*.

## List buckets

The following code example shows how to list S3 buckets.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::ListBuckets(const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    }
    else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << " buckets
\n";
```

```
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }

    return result;
}
```

- For API details, see [ListBuckets](#) in *AWS SDK for C++ API Reference*.

## List objects in a bucket

The following code example shows how to list objects in an S3 bucket.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::ListObjects(const Aws::String &bucketName,
                             const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::ListObjectsRequest request;
    request.WithBucket(bucketName);

    auto outcome = s3_client.ListObjects(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ListObjects: " <<
                    outcome.GetError().GetMessage() << std::endl;
    }
    else {
        Aws::Vector<Aws::S3::Model::Object> objects =
            outcome.GetResult().GetContents();

        for (Aws::S3::Model::Object &object: objects) {
            std::cout << object.GetKey() << std::endl;
        }
    }
}

return outcome.IsSuccess();
}
```

- For API details, see [ListObjects](#) in *AWS SDK for C++ API Reference*.

## Set the website configuration for a bucket

The following code example shows how to set the website configuration for an S3 bucket.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::PutWebsiteConfig(const Aws::String &bucketName,
                                   const Aws::String &indexPage, const Aws::String
                                   &errorPage,
                                   const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
        client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
              << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Set website configuration for bucket '"
              << bucketName << '.' << std::endl;
    }
}

return outcome.IsSuccess();
}
```

- For API details, see [PutBucketWebsite](#) in *AWS SDK for C++ API Reference*.

## Upload an object to a bucket

The following code example shows how to upload an object to an S3 bucket.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::S3::PutObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
```

```
Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                               fileName.c_str(),
                               std::ios_base::in | std::ios_base::binary);

if (!*inputData) {
    std::cerr << "Error unable to read file " << fileName << std::endl;
    return false;
}

request.SetBody(inputData);

Aws::S3::Model::PutObjectOutcome outcome =
    s3_client.PutObject(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: PutObject: " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Added object '" << fileName << "' to bucket ''"
        << bucketName << "'.";
}

return outcome.IsSuccess();
}
```

- For API details, see [PutObject](#) in *AWS SDK for C++ API Reference*.

## Scenarios

### Get started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.
- Delete a bucket.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
```

```
#include <aws/s3/model/ListObjectsRequest.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <aws/core/utils/memory/stl/AWSStreamFwd.h>
#include <fstream>
#include "awsdoc/s3/s3_examples.h"

namespace AwsDoc {
    namespace S3 {

        //! Delete an S3 bucket.
        /*!
         \sa DeleteBucket()
         \param bucketName The S3 bucket's name.
         \param client An S3 client.
        */
        static bool DeleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

        //! Delete an object in an S3 bucket.
        /*!
         \sa DeleteObjectFromBucket()
         \param bucketName The S3 bucket's name.
         \param key The key for the object in the S3 bucket.
         \param client An S3 client.
        */
        static bool DeleteObjectFromBucket(const Aws::String &bucketName, const Aws::String &key, Aws::S3::S3Client &client);
    }
}

/// Scenario to create, copy, and delete S3 buckets and objects.
/*!
 \sa S3_GettingStartedScenario()
 \param uploadFilePath Path to file to upload to an Amazon S3 bucket.
 \param saveFilePath Path for saving a downloaded S3 object.
 \param clientConfig Aws client configuration.
*/
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &uploadFilePath, const Aws::String &saveFilePath,
                                            const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: "doc-example-bucket-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String bucketName = "doc-example-bucket-" +
                           Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
            createBucketConfiguration.WithLocationConstraint(
                Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(

```

```
        clientConfig.region));
    request.WithCreateBucketConfiguration(createBucketConfiguration);
}

Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);

if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &err = outcome.GetError();
    std::cerr << "Error: CreateBucket: " <<
        err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
    return false;
}
else {
    std::cout << "Created the bucket, '" << bucketName <<
        "', in the region, '" << clientConfig.region << "." <<
std::endl;
}
}

// 2. Upload a local file to the bucket.
Aws::String key = "key-for-test";
{
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    std::shared_ptr<Aws::FStream> input_data =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                       uploadFilePath,
                                       std::ios_base::in |
std::ios_base::binary);

    if (!input_data->is_open()) {
        std::cerr << "Error: unable to open file, '" << uploadFilePath << "'." <<
std::endl;
        AwsDoc::S3::DeleteBucket(bucketName, client);
        return false;
    }

    request.SetBody(input_data);

    Aws::S3::Model::PutObjectOutcome outcome =
        client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutObject: " <<
            outcome.GetError().GetMessage() << std::endl;
        AwsDoc::S3::DeleteObjectFromBucket(bucketName, key, client);
        AwsDoc::S3::DeleteBucket(bucketName, client);
        return false;
    }
    else {
        std::cout << "Added the object with the key, '" << key << "', to the
bucket, '" <<
            bucketName << "." << std::endl;
    }
}

// 3. Download the object to a local file.
{
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);
```

```

        if (!outcome.IsSuccess()) {
            const Aws::S3::S3Error &err = outcome.GetError();
            std::cerr << "Error: GetObject: " <<
                err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
        }
        else {
            std::cout << "Downloaded the object with the key, '" << key << "', in the
            bucket, '''
                            << bucketName << "." << std::endl;

            Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
                GetBody();
            Aws::OFStream outStream(saveFilePath);
            if (!outStream.is_open()) {
                std::cout << "Error: unable to open file, '" << saveFilePath << "'." <<
            std::endl;
            }
            else {
                outStream << ioStream.rdbuf();
                std::cout << "Wrote the downloaded object to the file ''"
                            << saveFilePath << "." << std::endl;
            }
        }
    }

    // 4. Copy the object to a different "folder" in the bucket.
    Aws::String copiedToKey = "test-folder/" + key;
    {
        Aws::S3::Model::CopyObjectRequest request;
        request.WithBucket(bucketName)
            .WithKey(copiedToKey)
            .WithCopySource(bucketName + "/" + key);

        Aws::S3::Model::CopyObjectOutcome outcome =
            client.CopyObject(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error: CopyObject: " <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else {
            std::cout << "Copied the object with the key, '" << key << "', to the key,
            '" << copiedToKey
                            << ", in the bucket, '" << bucketName << "." << std::endl;
        }
    }

    // 5. List objects in the bucket.
    {
        Aws::S3::Model::ListObjectsRequest request;
        request.WithBucket(bucketName);

        Aws::S3::Model::ListObjectsOutcome outcome = client.ListObjects(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: ListObjects: " <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                outcome.GetResult().GetContents();

            std::cout << objects.size() << " objects in the bucket, '" << bucketName <<
            "'." << std::endl;

            for (Aws::S3::Model::Object &object: objects) {

```

```

        std::cout << "      '" << object.GetKey() << "'" << std::endl;
    }
}

// 6. Delete all objects in the bucket.
// All objects in the bucket must be deleted before deleting the bucket.
AwsDoc::S3::DeleteObjectFromBucket(bucketName, copiedToKey, client);
AwsDoc::S3::DeleteObjectFromBucket(bucketName, key, client);

// 7. Delete the bucket.
return AwsDoc::S3::DeleteBucket(bucketName, client);
}

bool AwsDoc::S3::DeleteObjectFromBucket(const Aws::String &bucketName, const
                                         Aws::String &key,
                                         Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: DeleteObject: " <<
        outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Deleted the object with the key, '" << key << "' , from the
        bucket, ''"
                    << bucketName << "." << std::endl;
    }
    return outcome.IsSuccess();
}

bool AwsDoc::S3::DeleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client)
{
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: DeleteBucket: " <<
        err.GetExceptionName() << ":" << err.GetMessage() << std::endl;
    }
    else {
        std::cout << "Deleted the bucket, '" << bucketName << "." << std::endl;
    }
    return outcome.IsSuccess();
}

```

- For API details, see the following topics in *AWS SDK for C++ API Reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)

- [ListObjects](#)
- [PutObject](#)

## Amazon SNS examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon Simple Notification Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2698\)](#)

## Actions

### Create a topic

The following code example shows how to create an Amazon SNS topic.

#### SDK for C++

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::String topic_name = argv[1];
    Aws::SNS::SNSClient sns;

    Aws::SNS::Model::CreateTopicRequest ct_req;
    ct_req.SetName(topic_name);

    auto ct_out = sns.CreateTopic(ct_req);

    if (ct_out.IsSuccess())
    {
        std::cout << "Successfully created topic " << topic_name << std::endl;
    }
    else
    {
        std::cout << "Error creating topic " << topic_name << ":" <<
        ct_out.GetError().GetMessage() << std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [CreateTopic](#) in *AWS SDK for C++ API Reference*.

### Delete a subscription

The following code example shows how to delete an Amazon SNS subscription.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;
    Aws::String subscription_arn = argv[1];

    Aws::SNS::Model::UnsubscribeRequest s_req;
    s_req.SetSubscriptionArn(subscription_arn);

    auto s_out = sns.Unsubscribe(s_req);

    if (s_out.IsSuccess())
    {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else
    {
        std::cout << "Error while unsubscribing " << s_out.GetError().GetMessage()
            << std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [Unsubscribe](#) in *AWS SDK for C++ API Reference*.

## Delete a topic

The following code example shows how to delete an Amazon SNS topic and all subscriptions to that topic.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::String topic_arn = argv[1];
    Aws::SNS::SNSClient sns;

    Aws::SNS::Model::DeleteTopicRequest dt_req;
    dt_req.SetTopicArn(topic_arn);

    auto dt_out = sns.DeleteTopic(dt_req);

    if (dt_out.IsSuccess())
    {
        std::cout << "Successfully deleted topic " << topic_arn << std::endl;
    }
    else
    {
        std::cout << "Error deleting topic " << topic_arn << ":" <<
```

```
        dt_out.GetError().GetMessage() << std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [DeleteTopic](#) in *AWS SDK for C++ API Reference*.

## Get the properties of a topic

The following code example shows how to get the properties of an Amazon SNS topic.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;
    Aws::String topic_arn = argv[1];

    Aws::SNS::Model::GetTopicAttributesRequest gta_req;
    gta_req.SetTopicArn(topic_arn);

    auto gta_out = sns.GetTopicAttributes(gta_req);

    if (gta_out.IsSuccess())
    {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute : gta_out.GetResult().GetAttributes())
        {
            std::cout << " * " << attribute.first << " : " << attribute.second <<
std::endl;
        }
    }
    else
    {
        std::cout << "Error while getting Topic attributes " <<
gta_out.GetError().GetMessage()
        << std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for C++ API Reference*.

## Get the settings for sending SMS messages

The following code example shows how to get the settings for sending Amazon SNS SMS messages.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;

    Aws::SNS::Model::GetSMSAttributesRequest gsmst_req;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    gsmst_req.AddAttributes("DefaultSMSType");

    auto gsmst_out = sns.GetSMSAttributes(gsmst_req);

    if (gsmst_out.IsSuccess())
    {
        for (auto const& att : gsmst_out.GetResult().GetAttributes())
        {
            std::cout << att.first << ":" << att.second << std::endl;
        }
    }
    else
    {
        std::cout << "Error while getting SMS Type: '" <<
        gsmst_out.GetError().GetMessage()
        << "'" << std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [GetSMSAttributes](#) in *AWS SDK for C++ API Reference*.

## List the subscribers of a topic

The following code example shows how to retrieve the list of subscribers of an Amazon SNS topic.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;

    Aws::SNS::Model::ListSubscriptionsRequest ls_req;

    auto ls_out = sns.ListSubscriptions(ls_req);

    if (ls_out.IsSuccess())
    {
        std::cout << "Subscriptions list:" << std::endl;
        for (auto const& subscription : ls_out.GetResult().GetSubscriptions())
        {
            std::cout << " * " << subscription.GetSubscriptionArn() << std::endl;
        }
    }
    else
    {
```

```
        std::cout << "Error listing subscriptions " << ls_out.GetError().GetMessage() <<
        std::endl;
    }

Aws::ShutdownAPI(options);
```

- For API details, see [ListSubscriptions](#) in *AWS SDK for C++ API Reference*.

## List topics

The following code example shows how to list Amazon SNS topics.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;

    Aws::SNS::Model::ListTopicsRequest lt_req;

    auto lt_out = sns.ListTopics(lt_req);

    if (lt_out.IsSuccess())
    {
        std::cout << "Topics list:" << std::endl;
        for (auto const &topic : lt_out.GetResult().GetTopics())
        {
            std::cout << " * " << topic.GetTopicArn() << std::endl;
        }
    }
    else
    {
        std::cout << "Error listing topics " << lt_out.GetError().GetMessage() <<
        std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [ListTopics](#) in *AWS SDK for C++ API Reference*.

## Publish an SMS text message

The following code example shows how to publish SMS messages using Amazon SNS.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
```

```
* Publish SMS: use Amazon SNS to send an SMS text message to a phone number.
* Note: This requires additional AWS configuration prior to running example.
*
* NOTE: When you start using Amazon SNS to send SMS messages, your AWS account is in
the SMS sandbox and you can only
* use verified destination phone numbers. See https://docs.aws.amazon.com/sns/latest/dg/sns-sms-sandbox.html.
* NOTE: If destination is in the US, you also have an additional restriction that you
have use a dedicated
* origination ID (phone number). You can request an origination number using Amazon
Pinpoint for a fee.
* See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-origination-numbers-with-amazon-sns/
* for more information.
*
* <phone_number_value> input parameter uses E.164 format.
* For example, in United States, this input value should be of the form: +12223334444
*/
int main(int argc, char ** argv)
{
    if (argc != 3)
    {
        std::cout << "Usage: publish_sms <message_value> <phone_number_value> " <<
std::endl;
        return 1;
    }

    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        Aws::SNS::SNSClient sns;
        Aws::String message = argv[1];
        Aws::String phone_number = argv[2];

        Aws::SNS::Model::PublishRequest psms_req;
        psms_req.SetMessage(message);
        psms_req.SetPhoneNumber(phone_number);

        auto psms_out = sns.Publish(psms_req);

        if (psms_out.IsSuccess())
        {
            std::cout << "Message published successfully " <<
psms_out.GetResult().GetMessageId()
                << std::endl;
        }
        else
        {
            std::cout << "Error while publishing message " <<
psms_out.GetError().GetMessage()
                << std::endl;
        }
    }

    Aws::ShutdownAPI(options);
    return 0;
}
```

- For API details, see [Publish](#) in [AWS SDK for C++ API Reference](#).

## [Publish to a topic](#)

The following code example shows how to publish messages to an Amazon SNS topic.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;
    Aws::String message = argv[1];
    Aws::String topic_arn = argv[2];

    Aws::SNS::Model::PublishRequest psms_req;
    psms_req.SetMessage(message);
    psms_req.SetTopicArn(topic_arn);

    auto psms_out = sns.Publish(psms_req);

    if (psms_out.IsSuccess())
    {
        std::cout << "Message published successfully " << std::endl;
    }
    else
    {
        std::cout << "Error while publishing message " <<
        psms_out.GetError().GetMessage()
                    << std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [Publish](#) in *AWS SDK for C++ API Reference*.

## Set the default settings for sending SMS messages

The following code example shows how to set the default settings for sending SMS messages using Amazon SNS.

## SDK for C++

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

How to use Amazon SNS to set the DefaultSMSType attribute.

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;
    Aws::String sms_type = argv[1];

    Aws::SNS::Model::SetSMSAttributesRequest ssmst_req;
    ssmst_req.AddAttributes("DefaultSMSType", sms_type);

    auto ssmst_out = sns.SetSMSAttributes(ssmst_req);

    if (ssmst_out.IsSuccess())
    {
```

```
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else
    {
        std::cout << "Error while setting SMS Type: '" <<
ssmst_out.GetError().GetMessage()
                << "'" << std::endl;
    }
}

Aws::ShutdownAPI(options);
```

- For API details, see [SetSmsAttributes](#) in [AWS SDK for C++ API Reference](#).

## Subscribe a Lambda function to a topic

The following code example shows how to subscribe a Lambda function so it receives notifications from an Amazon SNS topic.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/** 
 * Subscribe an AWS Lambda endpoint to a topic - demonstrates how to initiate a
subscription to an Amazon SNS topic with delivery
 * to an AWS Lambda function.
 *
 * NOTE: You must first configure AWS Lambda to run this example.
 *       See https://docs.amazonaws.cn/en\_us/lambda/latest/dg/with-sns-example.html
for more information.
*
* <protocol_value> set to "lambda" provides delivery of JSON-encoded message to an AWS
Lambda function
*       (see https://docs.aws.amazon.com/sns/latest/api/API\_Subscribe.html for
available protocols).
* <topic_arn_value> can be obtained from run_list_topics executable and includes the
"arn:" prefix.
* <lambd_function_arn> is the ARN of an AWS Lambda function.
*/
int main(int argc, char ** argv)
{
    if (argc != 4)
    {
        std::cout << "Usage: subscribe_lambda <protocol_value=lambda> <topic_arn_value>" 
                    " <lambd_function_arn>" << std::endl;
        return 1;
    }

    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        Aws::SNS::SNSClient sns;
        Aws::String protocol = argv[1];
        Aws::String topic_arn = argv[2];
        Aws::String endpoint = argv[3];

        Aws::SNS::Model::SubscribeRequest s_req;
        s_req.SetTopicArn(topic_arn);
```

```
s_req.SetProtocol(protocol);
s_req.SetEndpoint(endpoint);

auto s_out = sns.Subscribe(s_req);

if (s_out.IsSuccess())
{
    std::cout << "Subscribed successfully " << std::endl;
}
else
{
    std::cout << "Error while subscribing " << s_out.GetError().GetMessage()
        << std::endl;
}
}

Aws::ShutdownAPI(options);
return 0;
}
```

- For API details, see [Subscribe](#) in *AWS SDK for C++ API Reference*.

## Subscribe a mobile application to a topic

The following code example shows how to subscribe a mobile application endpoint so it receives notifications from an Amazon SNS topic.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
 * Subscribe an app endpoint to a topic - demonstrates how to initiate a subscription
 * to an Amazon SNS topic
 * with delivery to a mobile app.
 *
 * NOTE: You must first create an endpoint by registering an app and device.
 *       See https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-devicetoken.html
 * for more information.
 *
 * <protocol_value> set to "application" provides delivery of JSON-encoded message to
 * an EndpointArn
 *         for a mobile app and device (see https://docs.aws.amazon.com/sns/latest/api/API\_Subscribe.html for available protocols).
 * <topic_arn_value> can be obtained from run_list_topics executable and includes the
 * "arn:" prefix.
 * <mobile_endpoint_arn> is the EndpointArn of a mobile app and device.
 */
int main(int argc, char ** argv)
{
    if (argc != 4)
    {
        std::cout << "Usage: subscribe_app <protocol_value/application> <topic_arn_value>"
            " <mobile_endpoint_arn>" << std::endl;
        return 1;
    }

    Aws::SDKOptions options;
    Aws::InitAPI(options);
{
```

```
Aws::SNS::SNSClient sns;
Aws::String protocol = argv[1];
Aws::String topic_arn = argv[2];
Aws::String endpoint = argv[3];

Aws::SNS::Model::SubscribeRequest s_req;
s_req.SetTopicArn(topic_arn);
s_req.SetProtocol(protocol);
s_req.SetEndpoint(endpoint);

auto s_out = sns.Subscribe(s_req);

if (s_out.IsSuccess())
{
    std::cout << "Subscribed successfully " << std::endl;
}
else
{
    std::cout << "Error while subscribing " << s_out.GetError().GetMessage()
        << std::endl;
}
}

Aws::ShutdownAPI(options);
return 0;
}
```

- For API details, see [Subscribe](#) in *AWS SDK for C++ API Reference*.

## Subscribe an email address to a topic

The following code example shows how to subscribe an email address to an Amazon SNS topic.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
 * Subscribe an email address endpoint to a topic - demonstrates how to initiate a
 * subscription to an Amazon SNS topic with delivery
 * to an email address.
 *
 * SNS will send a subscription confirmation email to the email address provided which
 * you need to confirm to
 * receive messages.
 *
 * <protocol_value> set to "email" provides delivery of message via SMTP (see https://
docs.aws.amazon.com/sns/latest/api/API_Subscribe.html for available protocols).
 * <topic_arn_value> can be obtained from run_list_topics executable and includes the
 "arn:" prefix.
 */

int main(int argc, char ** argv)
{
    if (argc != 4)
    {
        std::cout << "Usage: subscribe_email <protocol_value=email> <topic_arn_value>"
            " <email_address>" << std::endl;
        return 1;
    }
}
```

```
Aws::SDKOptions options;
Aws::InitAPI(options);
{
    Aws::SNS::SNSClient sns;
    Aws::String protocol = argv[1];
    Aws::String topic_arn = argv[2];
    Aws::String endpoint = argv[3];

    Aws::SNS::Model::SubscribeRequest s_req;
    s_req.SetTopicArn(topic_arn);
    s_req.SetProtocol(protocol);
    s_req.SetEndpoint(endpoint);

    auto s_out = sns.Subscribe(s_req);

    if (s_out.IsSuccess())
    {
        std::cout << "Subscribed successfully " << std::endl;
    }
    else
    {
        std::cout << "Error while subscribing " << s_out.GetError().GetMessage()
            << std::endl;
    }
}

Aws::ShutdownAPI(options);
return 0;
}
```

- For API details, see [Subscribe](#) in *AWS SDK for C++ API Reference*.

## AWS STS examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with AWS Security Token Service.

**Actions** are code excerpts that show you how to call individual service functions.

**Scenarios** are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2708\)](#)

## Actions

### Assume a role

The following code example shows how to assume a role with AWS STS.

#### SDK for C++

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
```

```
        const Aws::String &roleSessionName,
        const Aws::String &externalId,
        Aws::Auth::AWSCredentials &credentials,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::STS::STSService sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials = result.GetCredentials();

        // Store temporary credentials in return argument.
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }
}

return outcome.IsSuccess();
}
```

- For API details, see [AssumeRole](#) in *AWS SDK for C++ API Reference*.

## Secrets Manager examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with AWS Secrets Manager.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2709\)](#)

## Actions

### Create a secret

The following code example shows how to create a Secrets Manager secret.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
int main(int argc, const char *argv[])
{
    if (argc != 3) {
        std::cout << "Usage:\n" <<
        "    <secretName> <secretValue> \n\n" <<
        "Where:\n" <<
        "    secretName - The name of the secret (for example, tutorials/
MyFirstSecret). \n" <<
        "    secretValue - The secret value. " << std::endl;
        return 0;
    }

    SDKOptions options;
    options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;

    InitAPI(options);
    {
        Aws::Client::ClientConfiguration config;

        //TODO(user): Enter the Region where you want to create the secret.
        String region = "us-east-1";
        if (!region.empty())
        {
            config.region = region;
        }
        SecretsManager::SecretsManagerClient sm_client(config);

        String secretName = argv[1];
        String secretString = argv[2];
        SecretsManager::Model::CreateSecretRequest request;
        request.SetName(secretName);
        request.SetSecretString(secretString);

        auto createSecretOutcome = sm_client.CreateSecret(request);
        if(createSecretOutcome.IsSuccess()){
            std::cout << "Create secret with name: " <<
createSecretOutcome.GetResult().GetName() << std::endl;
        }else{
            std::cout << "Failed with Error: " << createSecretOutcome.GetError() <<
std::endl;
        }
    }

    ShutdownAPI(options);
    return 0;
}
```

- For API details, see [CreateSecret](#) in *AWS SDK for C++ API Reference*.

## Get a secret value

The following code example shows how to get a Secrets Manager secret value.

### SDK for C++

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
int main(int argc, const char *argv[])
```

```
{  
    if (argc != 2) {  
        std::cout << "Usage:\n" <<  
            "    <secretName> \n\n" <<  
            "Where:\n" <<  
            "    secretName - The name of the secret (for example, tutorials/  
MyFirstSecret). \n"  
        << std::endl;  
        return 0;  
    }  
  
    SDKOptions options;  
    options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;  
  
    InitAPI(options);  
    {  
        Aws::Client::ClientConfiguration config;  
  
        //TODO(user): Enter the Region where you want to create the secret.  
        String region = "us-east-1";  
        if (!region.empty())  
        {  
            config.region = region;  
        }  
        SecretsManager::SecretsManagerClient sm_client(config);  
  
        String secretId = argv[1];  
        SecretsManager::Model::GetSecretValueRequest request;  
        request.SetSecretId(secretId);  
  
        auto getSecretValueOutcome = sm_client.GetSecretValue(request);  
        if(getSecretValueOutcome.IsSuccess()){  
            std::cout << "Secret is: " <<  
getSecretValueOutcome.GetResult().GetSecretString() << std::endl;  
        }else{  
            std::cout << "Failed with Error: " << getSecretValueOutcome.GetError()  
<< std::endl;  
        }  
    }  
  
    ShutdownAPI(options);  
    return 0;  
}
```

- For API details, see [GetSecretValue in AWS SDK for C++ API Reference](#).

## Amazon Transcribe examples using SDK for C++

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for C++ with Amazon Transcribe.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2712\)](#)

## Actions

### Produce real-time transcriptions

The following code example shows how to produce a real-time transcription with Amazon Transcribe.

#### SDK for C++

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
int main() {
    Aws::SDKOptions options;
    options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;
    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;

        Aws::Utils::Threading::Semaphore canCloseStream(0 /*initialCount*/, 1 /
*maxCount*/);

        //Load a profile that has been granted AmazonTranscribeFullAccess AWS managed
        //permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with AWS SDK version 1.9, this example only runs if
        // the SDK is built
        // with the curl library. (9/15/2022)
        // For more information, see the accompanying ReadMe.
        // For more information, see "Building the SDK for Windows with curl".
        // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
windows.html
        //TODO(User): Update to the location of your .crt file.
        config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
        config.region = region;

        TranscribeStreamingServiceClient client(config);
        StartStreamTranscriptionHandler handler;
        handler.SetOnErrorCallback(
            [](const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
&error) {
                std::cerr << "ERROR: " + error.GetMessage() << std::endl;
            });
        //SetTranscriptEventCallback called for every 'chunk' of file transcribed.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([&canCloseStream](const TranscriptEvent &ev)
{
            bool isFinal = false;
            for (auto &&r: ev.GetTranscript().GetResults()) {
                if (r.GetIsPartial()) {
                    std::cout << "[partial] ";
                }
                else {
                    std::cout << "[Final] ";
                    isFinal = true;
                }
                for (auto &&alt: r.GetAlternatives()) {
                    std::cout << alt.GetTranscript() << std::endl;
                }
            }
        });
    }
}
```

```

        if (isFinal) {
            canCloseStream.Release();
        }
    });

StartStreamTranscriptionRequest request;
request.SetMediaSampleRateHertz(8000);
request.SetLanguageCode(LanguageCode::en_US);
request.SetMediaEncoding(
    MediaEncoding::pcm); // wav and aiff files are PCM formats.
request.SetEventStreamHandler(handler);

auto OnStreamReady = [&canCloseStream](AudioStream &stream) {
    Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
    if (!file.is_open()) {
        std::cerr << "Failed to open " << FILE_NAME << '\n';
    }
    std::array<char, BUFFER_SIZE> buf;
    int i = 0;
    while (file) {
        file.read(&buf[0], buf.size());
        if (!file)
            std::cout << "File: only " << file.gcount() << " could be read"
            << std::endl;

        Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
        AudioEvent event(std::move(bits));
        if (!stream) {
            std::cerr << "Failed to create a stream" << std::endl;
            break;
        }
        //The std::basic_istream::gcount() is used to count the characters
        //in the given string. It returns
        //the number of characters extracted by the last read() operation.
        if (file.gcount() > 0) {
            if (!stream.WriteAudioEvent(event)) {
                std::cerr << "Failed to write an audio event" << std::endl;
                break;
            }
        }
        else {
            break;
        }
        std::this_thread::sleep_for(std::chrono::milliseconds(
            25)); // Slow down because we are streaming from a file.
    }
    if (!stream.WriteAudioEvent(
        AudioEvent())) {
        // Per the spec, we have to send an empty event (an event without a
        // payload) at the end.
        std::cerr << "Failed to send an empty frame" << std::endl;
    }
    else {
        std::cout << "Successfully sent the empty frame" << std::endl;
    }
    stream.flush();
    // Wait until the final transcript or an error is received.
    // Closing the stream prematurely will trigger an error.
    canCloseStream.WaitOne();
    stream.Close();
};

Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /*maxCount*/);
auto OnResponseCallback = [&signaling, &canCloseStream]()

```

```
    const TranscribeStreamingServiceClient * /*unused*/,
    const Model::StartStreamTranscriptionRequest & /*unused*/,
    const Model::StartStreamTranscriptionOutcome & outcome,
    const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /
*unused*/) {

    if (!outcome.IsSuccess())
    {
        std::cerr << "Transcribe streaming error " <<
outcome.GetError().GetMessage() << std::endl;
    }

    canCloseStream.Release();
    signaling.Release();
};

std::cout << "Starting..." << std::endl;
client.StartStreamTranscriptionAsync(request, OnStreamReady,
OnResponseCallback,
nullptr /*context*/);
signaling.WaitOne(); // Prevent the application from exiting until we're done.
std::cout << "Done" << std::endl;
}

Aws::ShutdownAPI(options);

return 0;
}
```

- For API details, see [StartStreamTranscriptionAsync](#) in *AWS SDK for C++ API Reference*.

## Cross-service examples using SDK for C++

The following sample applications use the AWS SDK for C++ to work across multiple AWS services.

### Examples

- [Create an Aurora Serverless work item tracker \(p. 2714\)](#)

## Create an Aurora Serverless work item tracker

### SDK for C++

Shows how to create a web application that tracks and reports on work items stored in an Amazon Aurora Serverless database.

For complete source code and instructions on how to set up a C++ REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

# Code examples for SDK for Go V2

The code examples in this topic show you how to use the AWS SDK for Go V2 with AWS.

## Examples

- [Single-service actions and scenarios using SDK for Go V2 \(p. 2715\)](#)

## Single-service actions and scenarios using SDK for Go V2

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for Go V2 with AWS services.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Services

- [DynamoDB examples using SDK for Go V2 \(p. 2715\)](#)
- [IAM examples using SDK for Go V2 \(p. 2746\)](#)
- [AWS KMS examples using SDK for Go V2 \(p. 2772\)](#)
- [Amazon S3 examples using SDK for Go V2 \(p. 2778\)](#)
- [Amazon SNS examples using SDK for Go V2 \(p. 2786\)](#)
- [Amazon SQS examples using SDK for Go V2 \(p. 2787\)](#)

## DynamoDB examples using SDK for Go V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Go V2 with Amazon DynamoDB.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Topics

- [Actions \(p. 2715\)](#)
- [Scenarios \(p. 2729\)](#)

## Actions

### Create a table

The following code example shows how to create a DynamoDB table.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the examples.  
// It contains a DynamoDB service client that is used to act on the specified table.  
type TableBasics struct {  
    dynamodb.Client  
    TableName string  
}  
  
// CreateMovieTable creates a DynamoDB table with a composite primary key defined as  
// a string sort key named `title`, and a numeric partition key named `year`.  
// This function uses NewTableExistsWaiter to wait for the table to be created by  
// DynamoDB before it returns.  
func (basics TableBasics) CreateMovieTable() (*types.TableDescription, error) {  
    var tableDesc *types.TableDescription  
    table, err := basics.DynamoDbClient.CreateTable(context.TODO(),  
&dynamodb.CreateTableInput{  
    AttributeDefinitions: []types.AttributeDefinition{  
        {AttributeName: aws.String("year"),  
         AttributeType: types.ScalarAttributeTypeN},  
    }, {  
        {AttributeName: aws.String("title"),  
         AttributeType: types.ScalarAttributeTypeS},  
    },  
    KeySchema: []types.KeySchemaElement{  
        {AttributeName: aws.String("year"),  
         KeyType: types.KeyTypeHash},  
    }, {  
        {AttributeName: aws.String("title"),  
         KeyType: types.KeyTypeRange},  
    },  
    TableName: aws.String(basics.TableName),  
    ProvisionedThroughput: &types.ProvisionedThroughput{  
        ReadCapacityUnits: aws.Int64(10),  
        WriteCapacityUnits: aws.Int64(10),  
    },  
})  
if err != nil {  
    log.Printf("Couldn't create table %v. Here's why: %v\n", basics.TableName, err)  
} else {  
    waiter := dynamodb.NewTableExistsWaiter(basics.DynamoDbClient)  
    err = waiter.Wait(context.TODO(), &dynamodb.DescribeTableInput{  
        TableName: aws.String(basics.TableName)}, 5*time.Minute)  
    if err != nil {  
        log.Printf("Wait for table exists failed. Here's why: %v\n", err)  
    }  
    tableDesc = table.TableDescription  
}  
return tableDesc, err  
}
```

- For API details, see [CreateTable](#) in *AWS SDK for Go API Reference*.

## Delete a table

The following code example shows how to delete a DynamoDB table.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// DeleteTable deletes the DynamoDB table and all of its data.
func (basics TableBasics) DeleteTable() error {
    _, err := basics.DynamoDbClient.DeleteTable(context.TODO(),
        &dynamodb.DeleteTableInput{
            TableName: aws.String(basics.TableName)})
    if err != nil {
        log.Printf("Couldn't delete table %v. Here's why: %v\n", basics.TableName, err)
    }
    return err
}
```

- For API details, see [DeleteTable](#) in *AWS SDK for Go API Reference*.

## Delete an item from a table

The following code example shows how to delete an item from a DynamoDB table.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// DeleteMovie removes a movie from the DynamoDB table.
func (basics TableBasics) DeleteMovie(movie Movie) error {
    _, err := basics.DynamoDbClient.DeleteItem(context.TODO(), &dynamodb.DeleteItemInput{
        TableName: aws.String(basics.TableName), Key: movie.GetKey(),
    })
    if err != nil {
        log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title, err)
    }
    return err
}
```

- For API details, see [DeleteItem](#) in *AWS SDK for Go API Reference*.

## Get an item from a table

The following code example shows how to get an item from a DynamoDB table.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the examples.  
// It contains a DynamoDB service client that is used to act on the specified table.  
type TableBasics struct {  
    dynamodb.Client  
    TableName string  
}  
  
  
// GetMovie gets movie data from the DynamoDB table by using the primary composite key  
// made of title and year.  
func (basics TableBasics) GetMovie(title string, year int) (Movie, error) {  
    movie := Movie{Title: title, Year: year}  
    response, err := basics.DynamoDbClient.GetItem(context.TODO(), &dynamodb.GetItemInput{  
        Key: movie.GetKey(), TableName: aws.String(basics.TableName),  
    })  
    if err != nil {  
        log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)  
    } else {  
        err = attributevalue.UnmarshalMap(response.Item, &movie)  
        if err != nil {  
            log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)  
        }  
    }  
    return movie, err  
}
```

- For API details, see [GetItem](#) in *AWS SDK for Go API Reference*.

## Get information about a table

The following code example shows how to get information about a DynamoDB table.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the examples.  
// It contains a DynamoDB service client that is used to act on the specified table.
```

```
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// TableExists determines whether a DynamoDB table exists.
func (basics TableBasics) TableExists() (bool, error) {
    exists := true
    _, err := basics.DynamoDbClient.DescribeTable(
        context.TODO(), &dynamodb.DescribeTableInput{TableName:
            aws.String(basics.TableName)},
    )
    if err != nil {
        var notFoundEx *types.ResourceNotFoundException
        if errors.As(err, &notFoundEx) {
            log.Printf("Table %v does not exist.\n", basics.TableName)
            err = nil
        } else {
            log.Printf("Couldn't determine existence of table %v. Here's why: %v\n",
                basics.TableName, err)
        }
        exists = false
    }
    return exists, err
}
```

- For API details, see [DescribeTable](#) in *AWS SDK for Go API Reference*.

## List tables

The following code example shows how to list DynamoDB tables.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// ListTables lists the DynamoDB table names for the current account.
func (basics TableBasics) ListTables() ([]string, error) {
    var tableNames []string
    tables, err := basics.DynamoDbClient.ListTables(
        context.TODO(), &dynamodb.ListTablesInput{})
    if err != nil {
        log.Printf("Couldn't list tables. Here's why: %v\n", err)
    } else {
        tableNames = tables.TableNames
    }
    return tableNames, err
}
```

```
}
```

- For API details, see [ListTables](#) in *AWS SDK for Go API Reference*.

## Put an item in a table

The following code example shows how to put an item in a DynamoDB table.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// AddMovie adds a movie to the DynamoDB table.
func (basics TableBasics) AddMovie(movie Movie) error {
    item, err := attributevalue.MarshalMap(movie)
    if err != nil {
        panic(err)
    }
    _, err = basics.DynamoDbClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
        TableName: aws.String(basics.TableName), Item: item,
    })
    if err != nil {
        log.Printf("Couldn't add item to table. Here's why: %v\n", err)
    }
    return err
}
```

- For API details, see [PutItem](#) in *AWS SDK for Go API Reference*.

## Query a table

The following code example shows how to query a DynamoDB table.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
}
```

```
    TableName      string
}

// Query gets all movies in the DynamoDB table that were released in the specified
// year.
// The function uses the `expression` package to build the key condition expression
// that is used in the query.
func (basics TableBasics) Query(releaseYear int) ([]Movie, error) {
    var err error
    var response *dynamodb.QueryOutput
    var movies []Movie
    keyEx := expression.Key("year").Equal(expression.Value(releaseYear))
    expr, err := expression.NewBuilder().WithKeyCondition(keyEx).Build()
    if err != nil {
        log.Printf("Couldn't build expression for query. Here's why: %v\n", err)
    } else {
        response, err = basics.DynamoDbClient.Query(context.TODO(), &dynamodb.QueryInput{
            TableName:           aws.String(basics.TableName),
            ExpressionAttributeNames: expr.Names(),
            ExpressionAttributeValues: expr.Values(),
            KeyConditionExpression:   expr.KeyCondition(),
        })
        if err != nil {
            log.Printf("Couldn't query for movies released in %v. Here's why: %v\n",
releaseYear, err)
        } else {
            err = attributevalue.UnmarshalListOfMaps(response.Items, &movies)
            if err != nil {
                log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
            }
        }
    }
    return movies, err
}
```

- For API details, see [Query in AWS SDK for Go API Reference](#).

## Run a PartiQL statement

The following code example shows how to run a PartiQL statement on a DynamoDB table.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use an INSERT statement to add an item.

```
// AddMovie runs a PartiQL INSERT statement to add a movie to the DynamoDB table.
func (runner PartiQLRunner) AddMovie(movie Movie) error {
    params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
    movie.Info})
    if err != nil {
        panic(err)
    }
    _, err = runner.DynamoDbClient.ExecuteStatement(context.TODO(),
&dynamodb.ExecuteStatementInput{
    Statement: aws.String(
```

```
    fmt.Sprintf("INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
    runner.TableName)),
Parameters: params,
})
if err != nil {
log.Printf("Couldn't insert an item with PartiQL. Here's why: %v\n", err)
}
return err
}
```

Use a SELECT statement to get an item.

```
// GetMovie runs a PartiQL SELECT statement to get a movie from the DynamoDB table by
// title and year.
func (runner PartiQLRunner) GetMovie(title string, year int) (Movie, error) {
var movie Movie
params, err := attributevalue.MarshalList([]interface{}{title, year})
if err != nil {
panic(err)
}
response, err := runner.DynamoDbClient.ExecuteStatement(context.TODO(),
&dynamodb.ExecuteStatementInput{
Statement: aws.String(
fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
runner.TableName)),
Parameters: params,
})
if err != nil {
log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
} else {
err = attributevalue.UnmarshalMap(response.Items[0], &movie)
if err != nil {
log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
}
}
return movie, err
}
```

Use a SELECT statement to get a list of items and project the results.

```
// GetAllMovies runs a PartiQL SELECT statement to get all movies from the DynamoDB
table.
// The results are projected to return only the title and rating of each movie.
func (runner PartiQLRunner) GetAllMovies() ([]map[string]interface{}, error) {
var output []map[string]interface{}
response, err := runner.DynamoDbClient.ExecuteStatement(context.TODO(),
&dynamodb.ExecuteStatementInput{
Statement: aws.String(
fmt.Sprintf("SELECT title, info.rating FROM \"%v\"", runner.TableName)),
})
if err != nil {
log.Printf("Couldn't get movies. Here's why: %v\n", err)
} else {
err = attributevalue.UnmarshalListOfMaps(response.Items, &output)
if err != nil {
log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
}
}
```

```
    }
    return output, err
}
```

Use an UPDATE statement to update an item.

```
// UpdateMovie runs a PartiQL UPDATE statement to update the rating of a movie that
// already exists in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovie(movie Movie, rating float64) error {
    params, err := attributevalue.MarshalList([]interface{}{rating, movie.Title,
        movie.Year})
    if err != nil {
        panic(err)
    }
    _, err = runner.DynamoDbClient.ExecuteStatement(context.TODO(),
        &dynamodb.ExecuteStatementInput{
            Statement: aws.String(
                fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
                    runner.TableName)),
            Parameters: params,
        })
    if err != nil {
        log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
    }
    return err
}
```

Use a DELETE statement to delete an item.

```
// DeleteMovie runs a PartiQL DELETE statement to remove a movie from the DynamoDB
// table.
func (runner PartiQLRunner) DeleteMovie(movie Movie) error {
    params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
    if err != nil {
        panic(err)
    }
    _, err = runner.DynamoDbClient.ExecuteStatement(context.TODO(),
        &dynamodb.ExecuteStatementInput{
            Statement: aws.String(
                fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
                    runner.TableName)),
            Parameters: params,
        })
    if err != nil {
        log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title, err)
    }
    return err
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Go API Reference*.

## Run batches of PartiQL statements

The following code example shows how to run batches of PartiQL statements on a DynamoDB table.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use batches of INSERT statements to add items.

```
// AddMovieBatch runs a batch of PartiQL INSERT statements to add multiple movies to
// the
// DynamoDB table.
func (runner PartiQLRunner) AddMovieBatch(movies []Movie) error {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
            movie.Info})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(fmt.Sprintf(
                "INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}", runner.TableName)),
            Parameters: params,
        }
    }

    _, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
        &dynamodb.BatchExecuteStatementInput{
            Statements: statementRequests,
        })
    if err != nil {
        log.Printf("Couldn't insert a batch of items with PartiQL. Here's why: %v\n", err)
    }
    return err
}
```

Use batches of SELECT statements to get items.

```
// GetMovieBatch runs a batch of PartiQL SELECT statements to get multiple movies from
// the DynamoDB table by title and year.
func (runner PartiQLRunner) GetMovieBatch(movies []Movie) ([]Movie, error) {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(
                fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
            Parameters: params,
        }
    }

    output, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
        &dynamodb.BatchExecuteStatementInput{
            Statements: statementRequests,
        })
    var outMovies []Movie
    if err != nil {
```

```
    log.Printf("Couldn't get a batch of items with PartiQL. Here's why: %v\n", err)
} else {
    for _, response := range output.Responses {
        var movie Movie
        err = attributevalue.UnmarshalMap(response.Item, &movie)
        if err != nil {
            log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
        } else {
            outMovies = append(outMovies, movie)
        }
    }
}
return outMovies, err
}
```

Use batches of UPDATE statements to update items.

```
// UpdateMovieBatch runs a batch of PartiQL UPDATE statements to update the rating of
// multiple movies that already exist in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovieBatch(movies []Movie, ratings []float64) error {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{ratings[index], movie.Title,
        movie.Year})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(
                fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
runner.TableName)),
            Parameters: params,
        }
    }

    _, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
&dynamodb.BatchExecuteStatementInput{
    Statements: statementRequests,
})
    if err != nil {
        log.Printf("Couldn't update the batch of movies. Here's why: %v\n", err)
    }
    return err
}
```

Use batches of DELETE statements to delete items.

```
// DeleteMovieBatch runs a batch of PartiQL DELETE statements to remove multiple movies
// from the DynamoDB table.
func (runner PartiQLRunner) DeleteMovieBatch(movies []Movie) error {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
```

```
    Statement: aws.String(
        fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
    Parameters: params,
}
}

_, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
&dynamodb.BatchExecuteStatementInput{
    Statements: statementRequests,
})
if err != nil {
    log.Printf("Couldn't delete the batch of movies. Here's why: %v\n", err)
}
return err
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Go API Reference*.

## Scan a table

The following code example shows how to scan a DynamoDB table.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// Scan gets all movies in the DynamoDB table that were released in a range of years
// and projects them to return a reduced set of fields.
// The function uses the `expression` package to build the filter and projection
// expressions.
func (basics TableBasics) Scan(startYear int, endYear int) ([]Movie, error) {
    var movies []Movie
    var err error
    var response *dynamodb.ScanOutput
    filtEx := expression.Name("year").Between(expression.Value(startYear),
        expression.Value(endYear))
    projEx := expression.NamesList(
        expression.Name("year"), expression.Name("title"), expression.Name("info.rating"))
    expr, err := expression.NewBuilder().WithFilter(filtEx).WithProjection(projEx).Build()
    if err != nil {
        log.Printf("Couldn't build expressions for scan. Here's why: %v\n", err)
    } else {
        response, err = basics.DynamoDbClient.Scan(context.TODO(), &dynamodb.ScanInput{
            TableName:                 aws.String(basics.TableName),
            ExpressionAttributeNames: expr.Names(),
            ExpressionAttributeValues: expr.Values(),
            FilterExpression:          expr.Filter(),
            ProjectionExpression:     expr.Projection(),
        })
    }
    if err != nil {
        log.Printf("Error scanning table: %v\n", err)
    } else {
        movies = response.Items
    }
    return movies, err
}
```

```
        })
        if err != nil {
            log.Printf("Couldn't scan for movies released between %v and %v. Here's why: %v\n",
                startYear, endYear, err)
        } else {
            err = attributevalue.UnmarshalListOfMaps(response.Items, &movies)
            if err != nil {
                log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
            }
        }
    }
    return movies, err
}
```

- For API details, see [Scan](#) in *AWS SDK for Go API Reference*.

## Update an item in a table

The following code example shows how to update an item in a DynamoDB table.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// UpdateMovie updates the rating and plot of a movie that already exists in the
// DynamoDB table. This function uses the `expression` package to build the update
// expression.
func (basics TableBasics) UpdateMovie(movie Movie) (map[string]map[string]interface{}, error) {
    var err error
    var response *dynamodb.UpdateItemOutput
    var attributeMap map[string]map[string]interface{}
    update := expression.Set(expression.Name("info.rating"),
        expression.Value(movie.Info["rating"]))
    update.Set(expression.Name("info.plot"), expression.Value(movie.Info["plot"]))
    expr, err := expression.NewBuilder().WithUpdate(update).Build()
    if err != nil {
        log.Printf("Couldn't build expression for update. Here's why: %v\n", err)
    } else {
        response, err = basics.DynamoDbClient.UpdateItem(context.TODO(),
            &dynamodb.UpdateItemInput{
                TableName:           aws.String(basics.TableName),
                Key:                movie.GetKey(),
                ExpressionAttributeNames: expr.Names(),
                ExpressionAttributeValues: expr.Values(),
                UpdateExpression:      expr.Update(),
                ReturnValue:          types.ReturnValueUpdatedNew,
            })
    }
}
```

```
    if err != nil {
        log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
    } else {
        err = attributevalue.UnmarshalMap(response.Attributes, &attributeMap)
        if err != nil {
            log.Printf("Couldn't unmarshal update response. Here's why: %v\n", err)
        }
    }
}
return attributeMap, err
}
```

- For API details, see [UpdateItem](#) in *AWS SDK for Go API Reference*.

## Write a batch of items

The following code example shows how to write a batch of DynamoDB items.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    dynamodb.Client
    TableName     string
}

// AddMovieBatch adds a slice of movies to the DynamoDB table. The function sends
// batches of 25 movies to DynamoDB until all movies are added or it reaches the
// specified maximum.
func (basics TableBasics) AddMovieBatch(movies []Movie, maxMovies int) (int, error) {
    var err error
    var item map[string]types.AttributeValue
    written := 0
    batchSize := 25 // DynamoDB allows a maximum batch size of 25 items.
    start := 0
    end := start + batchSize
    for start < maxMovies && start < len(movies) {
        var writeReqs []types.WriteRequest
        if end > len(movies) {
            end = len(movies)
        }
        for _, movie := range movies[start:end] {
            item, err = attributevalue.MarshalMap(movie)
            if err != nil {
                log.Printf("Couldn't marshal movie %v for batch writing. Here's why: %v\n",
                    movie.Title, err)
            } else {
                writeReqs = append(
                    writeReqs,
                    types.WriteRequest{PutRequest: &types.PutRequest{Item: item}},
                )
            }
        }
    }
}
```

```
_, err = basics.DynamoDbClient.BatchWriteItem(context.TODO(),
&dynamodb.BatchWriteItemInput{
    RequestItems: map[string][]types.WriteRequest{basics.TableName: writeReqs}})
if err != nil {
    log.Printf("Couldn't add a batch of movies to %v. Here's why: %v\n",
    basics.TableName, err)
} else {
    written += len(writeReqs)
}
start = end
end += batchSize
}

return written, err
}
```

- For API details, see [BatchWriteItem in AWS SDK for Go API Reference](#).

## Scenarios

### Get started using tables, items, and queries

The following code example shows how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.
- Delete a movie from the table.
- Delete the table.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a struct and methods that call DynamoDB actions.

```
// TableBasics encapsulates the Amazon DynamoDB service actions used in the examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// TableExists determines whether a DynamoDB table exists.
func (basics TableBasics) TableExists() (bool, error) {
    exists := true
    _, err := basics.DynamoDbClient.DescribeTable(
        context.TODO(), &dynamodb.DescribeTableInput{TableName:
            aws.String(basics.TableName)},
    )
}
```

```

if err != nil {
    var notFoundEx *types.ResourceNotFoundException
    if errors.As(err, &notFoundEx) {
        log.Printf("Table %v does not exist.\n", basics.TableName)
        err = nil
    } else {
        log.Printf("Couldn't determine existence of table %v. Here's why: %v\n",
        basics.TableName, err)
    }
    exists = false
}
return exists, err
}

// CreateMovieTable creates a DynamoDB table with a composite primary key defined as
// a string sort key named `title`, and a numeric partition key named `year`.
// This function uses NewTableExistsWaiter to wait for the table to be created by
// DynamoDB before it returns.
func (basics TableBasics) CreateMovieTable() (*types.TableDescription, error) {
    var tableDesc *types.TableDescription
    table, err := basics.DynamoDbClient.CreateTable(context.TODO(),
    &dynamodb.CreateTableInput{
        AttributeDefinitions: []types.AttributeDefinition{{
           AttributeName: aws.String("year"),
           AttributeType: types.ScalarAttributeTypeN,
        }, {
           AttributeName: aws.String("title"),
           AttributeType: types.ScalarAttributeTypeS,
        }},
        KeySchema: []types.KeySchemaElement{{
           AttributeName: aws.String("year"),
           KeyType:      types.KeyTypeHash,
        }, {
           AttributeName: aws.String("title"),
           KeyType:      types.KeyTypeRange,
        }},
        TableName: aws.String(basics.TableName),
        ProvisionedThroughput: &types.ProvisionedThroughput{
            ReadCapacityUnits: aws.Int64(10),
            WriteCapacityUnits: aws.Int64(10),
        },
    })
    if err != nil {
        log.Printf("Couldn't create table %v. Here's why: %v\n", basics.TableName, err)
    } else {
        waiter := dynamodb.NewTableExistsWaiter(basics.DynamoDbClient)
        err = waiter.Wait(context.TODO(), &dynamodb.DescribeTableInput{
            TableName: aws.String(basics.TableName)}, 5*time.Minute)
        if err != nil {
            log.Printf("Wait for table exists failed. Here's why: %v\n", err)
        }
        tableDesc = table.TableDescription
    }
    return tableDesc, err
}

// ListTables lists the DynamoDB table names for the current account.
func (basics TableBasics) ListTables() ([]string, error) {
    var tableNames []string
    tables, err := basics.DynamoDbClient.ListTables(
        context.TODO(), &dynamodb.ListTablesInput{})
    if err != nil {

```

```
    log.Printf("Couldn't list tables. Here's why: %v\n", err)
} else {
    tableNames = tables.TableNames
}
return tableNames, err
}

// AddMovie adds a movie to the DynamoDB table.
func (basics TableBasics) AddMovie(movie Movie) error {
    item, err := attributevalue.MarshalMap(movie)
    if err != nil {
        panic(err)
    }
    _, err = basics.DynamoDbClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
        TableName: aws.String(basics.TableName), Item: item,
    })
    if err != nil {
        log.Printf("Couldn't add item to table. Here's why: %v\n", err)
    }
    return err
}

// UpdateMovie updates the rating and plot of a movie that already exists in the
// DynamoDB table. This function uses the `expression` package to build the update
// expression.
func (basics TableBasics) UpdateMovie(movie Movie) (map[string]map[string]interface{}, error) {
    var err error
    var response *dynamodb.UpdateItemOutput
    var attributeMap map[string]map[string]interface{}
    update := expression.Set(expression.Name("info.rating"),
        expression.Value(movie.Info["rating"]))
    update.Set(expression.Name("info.plot"), expression.Value(movie.Info["plot"]))
    expr, err := expression.NewBuilder().WithUpdate(update).Build()
    if err != nil {
        log.Printf("Couldn't build expression for update. Here's why: %v\n", err)
    } else {
        response, err = basics.DynamoDbClient.UpdateItem(context.TODO(),
            &dynamodb.UpdateItemInput{
                TableName:                 aws.String(basics.TableName),
                Key:                      movie.GetKey(),
                ExpressionAttributeNames: expr.Names(),
                ExpressionAttributeValues: expr.Values(),
                UpdateExpression:          expr.Update(),
                ReturnValues:              types.ReturnValueUpdatedNew,
            })
        if err != nil {
            log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
        } else {
            err = attributevalue.UnmarshalMap(response.Attributes, &attributeMap)
            if err != nil {
                log.Printf("Couldn't unmarshall update response. Here's why: %v\n", err)
            }
        }
    }
    return attributeMap, err
}

// AddMovieBatch adds a slice of movies to the DynamoDB table. The function sends
// batches of 25 movies to DynamoDB until all movies are added or it reaches the
```

```
// specified maximum.
func (basics TableBasics) AddMovieBatch(movies []Movie, maxMovies int) (int, error) {
    var err error
    var item map[string]types.AttributeValue
    written := 0
    batchSize := 25 // DynamoDB allows a maximum batch size of 25 items.
    start := 0
    end := start + batchSize
    for start < maxMovies && start < len(movies) {
        var writeReqs []types.WriteRequest
        if end > len(movies) {
            end = len(movies)
        }
        for _, movie := range movies[start:end] {
            item, err = attributevalue.MarshalMap(movie)
            if err != nil {
                log.Printf("Couldn't marshal movie %v for batch writing. Here's why: %v\n",
                    movie.Title, err)
            } else {
                writeReqs = append(
                    writeReqs,
                    types.WriteRequest{PutRequest: &types.PutRequest{Item: item}},
                )
            }
        }
        _, err = basics.DynamoDbClient.BatchWriteItem(context.TODO(),
            &dynamodb.BatchWriteItemInput{
                RequestItems: map[string][]types.WriteRequest{basics.TableName: writeReqs}})
        if err != nil {
            log.Printf("Couldn't add a batch of movies to %v. Here's why: %v\n",
                basics.TableName, err)
        } else {
            written += len(writeReqs)
        }
        start = end
        end += batchSize
    }

    return written, err
}

// GetMovie gets movie data from the DynamoDB table by using the primary composite key
// made of title and year.
func (basics TableBasics) GetMovie(title string, year int) (Movie, error) {
    movie := Movie{Title: title, Year: year}
    response, err := basics.DynamoDbClient.GetItem(context.TODO(), &dynamodb.GetItemInput{
        Key: movie.GetKey(), TableName: aws.String(basics.TableName),
    })
    if err != nil {
        log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
    } else {
        err = attributevalue.UnmarshalMap(response.Item, &movie)
        if err != nil {
            log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
        }
    }
    return movie, err
}

// Query gets all movies in the DynamoDB table that were released in the specified
// year.
// The function uses the `expression` package to build the key condition expression
```

```
// that is used in the query.
func (basics TableBasics) Query(releaseYear int) ([]Movie, error) {
    var err error
    var response *dynamodb.QueryOutput
    var movies []Movie
    keyEx := expression.Key("year").Equal(expression.Value(releaseYear))
    expr, err := expression.NewBuilder().WithKeyCondition(keyEx).Build()
    if err != nil {
        log.Printf("Couldn't build expression for query. Here's why: %v\n", err)
    } else {
        response, err = basics.DynamoDbClient.Query(context.TODO(), &dynamodb.QueryInput{
            TableName:                 aws.String(basics.TableName),
            ExpressionAttributeNames: expr.Names(),
            ExpressionAttributeValues: expr.Values(),
            KeyConditionExpression:   expr.KeyCondition(),
        })
        if err != nil {
            log.Printf("Couldn't query for movies released in %v. Here's why: %v\n",
                releaseYear, err)
        } else {
            err = attributevalue.UnmarshalListOfMaps(response.Items, &movies)
            if err != nil {
                log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
            }
        }
    }
    return movies, err
}

// Scan gets all movies in the DynamoDB table that were released in a range of years
// and projects them to return a reduced set of fields.
// The function uses the `expression` package to build the filter and projection
// expressions.
func (basics TableBasics) Scan(startYear int, endYear int) ([]Movie, error) {
    var movies []Movie
    var err error
    var response *dynamodb.ScanOutput
    filtEx := expression.Name("year").Between(expression.Value(startYear),
        expression.Value(endYear))
    projEx := expression.NamesList(
        expression.Name("year"), expression.Name("title"), expression.Name("info.rating"))
    expr, err := expression.NewBuilder().WithFilter(filtEx).WithProjection(projEx).Build()
    if err != nil {
        log.Printf("Couldn't build expressions for scan. Here's why: %v\n", err)
    } else {
        response, err = basics.DynamoDbClient.Scan(context.TODO(), &dynamodb.ScanInput{
            TableName:                 aws.String(basics.TableName),
            ExpressionAttributeNames: expr.Names(),
            ExpressionAttributeValues: expr.Values(),
            FilterExpression:         expr.Filter(),
            ProjectionExpression:    expr.Projection(),
        })
        if err != nil {
            log.Printf("Couldn't scan for movies released between %v and %v. Here's why: %v\n",
                startYear, endYear, err)
        } else {
            err = attributevalue.UnmarshalListOfMaps(response.Items, &movies)
            if err != nil {
                log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
            }
        }
    }
    return movies, err
}
```

```
// DeleteMovie removes a movie from the DynamoDB table.
func (basics TableBasics) DeleteMovie(movie Movie) error {
    _, err := basics.DynamoDbClient.DeleteItem(context.TODO(), &dynamodb.DeleteItemInput{
        TableName: aws.String(basics.TableName), Key: movie.GetKey(),
    })
    if err != nil {
        log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title, err)
    }
    return err
}

// DeleteTable deletes the DynamoDB table and all of its data.
func (basics TableBasics) DeleteTable() error {
    _, err := basics.DynamoDbClient.DeleteTable(context.TODO(),
        &dynamodb.DeleteTableInput{
            TableName: aws.String(basics.TableName)})
    if err != nil {
        log.Printf("Couldn't delete table %v. Here's why: %v\n", basics.TableName, err)
    }
    return err
}
```

Run an interactive scenario to create the table and perform actions on it.

```
// RunMovieScenario is an interactive example that shows you how to use the AWS SDK for
// Go
// to create and use an Amazon DynamoDB table that stores data about movies.
//
// 1. Create a table that can hold movie data.
// 2. Put, get, and update a single movie in the table.
// 3. Write movie data to the table from a sample JSON file.
// 4. Query for movies that were released in a given year.
// 5. Scan for movies that were released in a range of years.
// 6. Delete a movie from the table.
// 7. Delete the table.
//
// This example creates a DynamoDB service client from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// The specified movie sampler is used to get sample data from a URL that is loaded
// into the named table.
func RunMovieScenario(
    sdkConfig aws.Config, questioner demotools.IQuestioner, tableName string,
    movieSampler actions.IMovieSampler) {
    defer func() {
        if r := recover(); r != nil {
            fmt.Printf("Something went wrong with the demo.")
        }
    }()
}

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon DynamoDB getting started demo.")
log.Println(strings.Repeat("-", 88))
```

```
tableBasics := actions.TableBasics{TableName: tableName,
    DynamoDbClient: dynamodb.NewFromConfig(sdkConfig)}

exists, err := tableBasics.TableExists()
if err != nil {
    panic(err)
}
if !exists {
    log.Printf("Creating table %v...\n", tableName)
    _, err = tableBasics.CreateMovieTable()
    if err != nil {
        panic(err)
    } else {
        log.Printf("Created table %v.\n", tableName)
    }
} else {
    log.Printf("Table %v already exists.\n", tableName)
}

var customMovie actions.Movie
customMovie.Title = questioner.Ask("Enter a movie title to add to the table:",
    []demotools.IAnswerValidator{demotools.NotEmpty{}})
customMovie.Year = questioner.AskInt("What year was it released?",
    []demotools.IAnswerValidator{demotoools.NotEmpty{}, demotools.InIntRange{
        Lower: 1900, Upper: 2030}})
customMovie.Info = map[string]interface{}{}
customMovie.Info["rating"] = questioner.AskFloat64(
    "Enter a rating between 1 and 10:", []demotools.IAnswerValidator{
        demotools.NotEmpty{}, demotools.InFloatRange{Lower: 1, Upper: 10}})
customMovie.Info["plot"] = questioner.Ask("What's the plot? ",
    []demotools.IAnswerValidator{demotools.NotEmpty{}})
err = tableBasics.AddMovie(customMovie)
if err == nil {
    log.Printf("Added %v to the movie table.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's update your movie. You previously rated it %v.\n",
    customMovie.Info["rating"])
customMovie.Info["rating"] = questioner.AskFloat64(
    "What new rating would you give it?", []demotools.IAnswerValidator{
        demotools.NotEmpty{}, demotools.InFloatRange{Lower: 1, Upper: 10}})
log.Printf("You summarized the plot as '%v'.\n", customMovie.Info["plot"])
customMovie.Info["plot"] = questioner.Ask("What would you say now?",
    []demotools.IAnswerValidator{demotools.NotEmpty{}})
attributes, err := tableBasics.UpdateMovie(customMovie)
if err == nil {
    log.Printf("Updated %v with new values.\n", customMovie.Title)
    for _, attVal := range attributes {
        for valKey, val := range attVal {
            log.Printf("\t%v: %v\n", valKey, val)
        }
    }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Getting movie data from %v and adding 250 movies to the table...\n",
    movieSampler.GetURL())
movies := movieSampler.GetSampleMovies()
written, err := tableBasics.AddMovieBatch(movies, 250)
if err != nil {
    panic(err)
} else {
    log.Printf("Added %v movies to the table.\n", written)
}
```

```

show := 10
if show > written {
    show = written
}
log.Printf("The first %v movies in the table are:", show)
for index, movie := range movies[:show] {
    log.Printf("\t%v. %v\n", index+1, movie.Title)
}
movieIndex := questioner.AskInt(
    "Enter the number of a movie to get info about it: ", []demotools.IAnswerValidator{
        demotools.InIntRange{Lower: 1, Upper: show}},
)
movie, err := tableBasics.GetMovie(movies[movieIndex-1].Title,
movies[movieIndex-1].Year)
if err == nil {
    log.Println(movie)
}
log.Println(strings.Repeat("-", 88))

log.Println("Let's get a list of movies released in a given year.")
releaseYear := questioner.AskInt("Enter a year between 1972 and 2018: ",
    []demotools.IAnswerValidator{demotoools.InIntRange{Lower: 1972, Upper: 2018}}},
releases, err := tableBasics.Query(releaseYear)
if err == nil {
    if len(releases) == 0 {
        log.Printf("I couldn't find any movies released in %v!\n", releaseYear)
    } else {
        for _, movie = range releases {
            log.Println(movie)
        }
    }
}
log.Println(strings.Repeat("-", 88))

log.Println("Now let's scan for movies released in a range of years.")
startYear := questioner.AskInt("Enter a year: ", []demotools.IAnswerValidator{
    demotools.InIntRange{Lower: 1972, Upper: 2018}})
endYear := questioner.AskInt("Enter another year: ", []demotools.IAnswerValidator{
    demotools.InIntRange{Lower: 1972, Upper: 2018}})
releases, err = tableBasics.Scan(startYear, endYear)
if err == nil {
    if len(releases) == 0 {
        log.Printf("I couldn't find any movies released between %v and %v!\n", startYear,
endYear)
    } else {
        log.Printf("Found %v movies. In this list, the plot is <nil> because "+
            "we used a projection expression when scanning for items to return only "+
            "the title, year, and rating.\n", len(releases))
        for _, movie = range releases {
            log.Println(movie)
        }
    }
}
log.Println(strings.Repeat("-", 88))

var tables []string
if questioner.AskBool("Do you want to list all of your tables? (y/n) ", "y") {
    tables, err = tableBasics.ListTables()
    if err == nil {
        log.Printf("Found %v tables:", len(tables))
        for _, table := range tables {
            log.Printf("\t%v", table)
        }
    }
}

```

```
log.Println(strings.Repeat("-", 88))

log.Printf("Let's remove your movie '%v'.\n", customMovie.Title)
if questioner.AskBool("Do you want to delete it from the table? (y/n) ", "y") {
    err = tableBasics.DeleteMovie(customMovie)
}
if err == nil {
    log.Printf("Deleted %v.\n", customMovie.Title)
}

if questioner.AskBool("Delete the table, too? (y/n)", "y") {
    err = tableBasics.DeleteTable()
} else {
    log.Println("Don't forget to delete the table when you're done or you might " +
        "incur charges on your account.")
}
if err == nil {
    log.Printf("Deleted table %v.\n", tableBasics.TableName)
}

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

- For API details, see the following topics in *AWS SDK for Go API Reference*.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## Query a table by using batches of PartiQL statements

The following code example shows how to:

- Get a batch of items by running multiple SELECT statements.
- Add a batch of items by running multiple INSERT statements.
- Update a batch of items by running multiple UPDATE statements.
- Delete a batch of items by running multiple DELETE statements.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a struct that is a receiver for methods that can run PartiQL statements.

```
// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on the
// specified table.
type PartiQLRunner struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// AddMovieBatch runs a batch of PartiQL INSERT statements to add multiple movies to
// the
// DynamoDB table.
func (runner PartiQLRunner) AddMovieBatch(movies []Movie) error {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
        movie.Info})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(fmt.Sprintf(
                "INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}", runner.TableName)),
            Parameters: params,
        }
    }
    _, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
    &dynamodb.BatchExecuteStatementInput{
        Statements: statementRequests,
    })
    if err != nil {
        log.Printf("Couldn't insert a batch of items with PartiQL. Here's why: %v\n", err)
    }
    return err
}

// GetMovieBatch runs a batch of PartiQL SELECT statements to get multiple movies from
// the DynamoDB table by title and year.
func (runner PartiQLRunner) GetMovieBatch(movies []Movie) ([]Movie, error) {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(
                fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
            Parameters: params,
        }
    }
    output, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
    &dynamodb.BatchExecuteStatementInput{
        Statements: statementRequests,
    })
    var outMovies []Movie
    if err != nil {
        log.Printf("Couldn't get a batch of items with PartiQL. Here's why: %v\n", err)
    } else {
        for _, response := range output.Responses {
            var movie Movie
```

```
    err = attributevalue.UnmarshalMap(response.Item, &movie)
    if err != nil {
        log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
    } else {
        outMovies = append(outMovies, movie)
    }
}
return outMovies, err
}

// GetAllMovies runs a PartiQL SELECT statement to get all movies from the DynamoDB
// table.
// The results are projected to return only the title and rating of each movie.
func (runner PartiQLRunner) GetAllMovies() ([]map[string]interface{}, error) {
    var output []map[string]interface{}
    response, err := runner.DynamoDbClient.ExecuteStatement(context.TODO(),
    &dynamodb.ExecuteStatementInput{
        Statement: aws.String(
            fmt.Sprintf("SELECT title, info.rating FROM \"%v\"", runner.TableName)),
    })
    if err != nil {
        log.Printf("Couldn't get movies. Here's why: %v\n", err)
    } else {
        err = attributevalue.UnmarshalListOfMaps(response.Items, &output)
        if err != nil {
            log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
        }
    }
    return output, err
}

// UpdateMovieBatch runs a batch of PartiQL UPDATE statements to update the rating of
// multiple movies that already exist in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovieBatch(movies []Movie, ratings []float64) error {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{ratings[index], movie.Title,
        movie.Year})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(
                fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
                runner.TableName)),
            Parameters: params,
        }
    }

    _, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
    &dynamodb.BatchExecuteStatementInput{
        Statements: statementRequests,
    })
    if err != nil {
        log.Printf("Couldn't update the batch of movies. Here's why: %v\n", err)
    }
    return err
}
```

```
// DeleteMovieBatch runs a batch of PartiQL DELETE statements to remove multiple movies
// from the DynamoDB table.
func (runner PartiQLRunner) DeleteMovieBatch(movies []Movie) error {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(
                fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
            Parameters: params,
        }
    }

    _, err := runner.DynamoDbClient.BatchExecuteStatement(context.TODO(),
        &dynamodb.BatchExecuteStatementInput{
            Statements: statementRequests,
        })
    if err != nil {
        log.Printf("Couldn't delete the batch of movies. Here's why: %v\n", err)
    }
    return err
}
```

Run a scenario that creates a table and runs batches of PartiQL queries.

```
// RunPartiQLBatchScenario shows you how to use the AWS SDK for Go
// to run batches of PartiQL statements to query a table that stores data about movies.
//
// * Use batches of PartiQL statements to add, get, update, and delete data for
//   individual movies.
//
// This example creates an Amazon DynamoDB service client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// This example creates and deletes a DynamoDB table to use during the scenario.
func RunPartiQLBatchScenario(sdkConfig aws.Config, tableName string) {
    defer func() {
        if r := recover(); r != nil {
            fmt.Printf("Something went wrong with the demo.")
        }
    }()
}

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon DynamoDB PartiQL batch demo.")
log.Println(strings.Repeat("-", 88))

tableBasics := actions.TableBasics{
    DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
    TableName:      tableName,
}
runner := actions.PartiQLRunner{
    DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
    TableName:      tableName,
}

exists, err := tableBasics.TableExists()
if err != nil {
    panic(err)
```

```

    }

    if !exists {
        log.Printf("Creating table %v...\n", tableName)
        _, err = tableBasics.CreateMovieTable()
        if err != nil {
            panic(err)
        } else {
            log.Printf("Created table %v.\n", tableName)
        }
    } else {
        log.Printf("Table %v already exists.\n", tableName)
    }
    log.Println(strings.Repeat("-", 88))

    currentYear, _, _ := time.Now().Date()
    customMovies := []actions.Movie{
        {
            Title: "House PartiQL",
            Year: currentYear - 5,
            Info: map[string]interface{}{
                "plot": "Wacky high jinks result from querying a mysterious database.",
                "rating": 8.5},
        },
        {
            Title: "House PartiQL 2",
            Year: currentYear - 3,
            Info: map[string]interface{}{
                "plot": "Moderate high jinks result from querying another mysterious database.",
                "rating": 6.5},
        },
        {
            Title: "House PartiQL 3",
            Year: currentYear - 1,
            Info: map[string]interface{}{
                "plot": "Tepid high jinks result from querying yet another mysterious database.",
                "rating": 2.5},
        },
    }

    log.Printf("Inserting a batch of movies into table '%v'.\n", tableName)
    err = runner.AddMovieBatch(customMovies)
    if err == nil {
        log.Printf("Added %v movies to the table.\n", len(customMovies))
    }
    log.Println(strings.Repeat("-", 88))

    log.Println("Getting data for a batch of movies.")
    movies, err := runner.GetMovieBatch(customMovies)
    if err == nil {
        for _, movie := range movies {
            log.Println(movie)
        }
    }
    log.Println(strings.Repeat("-", 88))

    newRatings := []float64{7.7, 4.4, 1.1}
    log.Println("Updating a batch of movies with new ratings.")
    err = runner.UpdateMovieBatch(customMovies, newRatings)
    if err == nil {
        log.Printf("Updated %v movies with new ratings.\n", len(customMovies))
    }
    log.Println(strings.Repeat("-", 88))

    log.Println("Getting projected data from the table to verify our update.")
    projections, err := runner.GetAllMovies()
    if err == nil {
        for _, projection := range projections {
            log.Println(projection)
        }
    }
    log.Println(strings.Repeat("-", 88))

```

```
log.Println("Deleting a batch of movies.")
err = runner.DeleteMovieBatch(customMovies)
if err == nil {
    log.Printf("Deleted %v movies.\n", len(customMovies))
}

err = tableBasics.DeleteTable()
if err == nil {
    log.Printf("Deleted table %v.\n", tableBasics.TableName)
}

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Go API Reference*.

## Query a table using PartiQL

The following code example shows how to:

- Get an item by running a SELECT statement.
- Add an item by running an INSERT statement.
- Update an item by running an UPDATE statement.
- Delete an item by running a DELETE statement.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a struct that is a receiver for methods that can run PartiQL statements.

```
// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on the
// specified table.
type PartiQLRunner struct {
    DynamoDbClient *dynamodb.Client
    TableName       string
}

// AddMovie runs a PartiQL INSERT statement to add a movie to the DynamoDB table.
func (runner PartiQLRunner) AddMovie(movie Movie) error {
    params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
        movie.Info})
    if err != nil {
        panic(err)
    }
    _, err = runner.DynamoDbClient.ExecuteStatement(context.TODO(),
        &dynamodb.ExecuteStatementInput{
            Statement: aws.String(
                fmt.Sprintf("INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
```

```
        runner.TableName)),
    Parameters: params,
})
if err != nil {
    log.Printf("Couldn't insert an item with PartiQL. Here's why: %v\n", err)
}
return err
}

// GetMovie runs a PartiQL SELECT statement to get a movie from the DynamoDB table by
// title and year.
func (runner PartiQLRunner) GetMovie(title string, year int) (Movie, error) {
var movie Movie
params, err := attributevalue.MarshalList([]interface{}{title, year})
if err != nil {
    panic(err)
}
response, err := runner.DynamoDbClient.ExecuteStatement(context.TODO(),
&dynamodb.ExecuteStatementInput{
    Statement: aws.String(
        fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?", 
            runner.TableName)),
    Parameters: params,
})
if err != nil {
    log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
} else {
    err = attributevalue.UnmarshalMap(response.Items[0], &movie)
    if err != nil {
        log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
    }
}
return movie, err
}

// UpdateMovie runs a PartiQL UPDATE statement to update the rating of a movie that
// already exists in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovie(movie Movie, rating float64) error {
params, err := attributevalue.MarshalList([]interface{}{rating, movie.Title, 
    movie.Year})
if err != nil {
    panic(err)
}
_, err = runner.DynamoDbClient.ExecuteStatement(context.TODO(),
&dynamodb.ExecuteStatementInput{
    Statement: aws.String(
        fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?", 
            runner.TableName)),
    Parameters: params,
})
if err != nil {
    log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
}
return err
}

// DeleteMovie runs a PartiQL DELETE statement to remove a movie from the DynamoDB
// table.
func (runner PartiQLRunner) DeleteMovie(movie Movie) error {
params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
```

```
if err != nil {
    panic(err)
}
_, err = runner.DynamoDbClient.ExecuteStatement(context.TODO(),
&dynamodb.ExecuteStatementInput{
    Statement: aws.String(
        fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
            runner.TableName)),
    Parameters: params,
})
if err != nil {
    log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title, err)
}
return err
}
```

Run a scenario that creates a table and runs PartiQL queries.

```
// RunPartiQLSingleScenario shows you how to use the AWS SDK for Go
// to use PartiQL to query a table that stores data about movies.
//
// * Use PartiQL statements to add, get, update, and delete data for individual movies.
//
// This example creates an Amazon DynamoDB service client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// This example creates and deletes a DynamoDB table to use during the scenario.
func RunPartiQLSingleScenario(sdkConfig aws.Config, tableName string) {
    defer func() {
        if r := recover(); r != nil {
            fmt.Printf("Something went wrong with the demo.")
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the Amazon DynamoDB PartiQL single action demo.")
    log.Println(strings.Repeat("-", 88))

    tableBasics := actions.TableBasics{
        DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
        TableName:      tableName,
    }
    runner := actions.PartiQLRunner{
        DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
        TableName:      tableName,
    }

    exists, err := tableBasics.TableExists()
    if err != nil {
        panic(err)
    }
    if !exists {
        log.Printf("Creating table %v...\n", tableName)
        _, err = tableBasics.CreateMovieTable()
        if err != nil {
            panic(err)
        } else {
            log.Printf("Created table %v.\n", tableName)
        }
    } else {
        log.Printf("Table %v already exists.\n", tableName)
```

```

    }

    log.Println(strings.Repeat("-", 88))

    currentYear, _, _ := time.Now().Date()
    customMovie := actions.Movie{
        Title: "24 Hour PartiQL People",
        Year: currentYear,
        Info: map[string]interface{}{
            "plot": "A group of data developers discover a new query language they can't stop
using.",
            "rating": 9.9,
        },
    }

    log.Printf("Inserting movie '%v' released in %v.", customMovie.Title,
    customMovie.Year)
    err = runner.AddMovie(customMovie)
    if err == nil {
        log.Printf("Added %v to the movie table.\n", customMovie.Title)
    }
    log.Println(strings.Repeat("-", 88))

    log.Printf("Getting data for movie '%v' released in %v.", customMovie.Title,
    customMovie.Year)
    movie, err := runner.GetMovie(customMovie.Title, customMovie.Year)
    if err == nil {
        log.Println(movie)
    }
    log.Println(strings.Repeat("-", 88))

    newRating := 6.6
    log.Printf("Updating movie '%v' with a rating of %v.", customMovie.Title, newRating)
    err = runner.UpdateMovie(customMovie, newRating)
    if err == nil {
        log.Printf("Updated %v with a new rating.\n", customMovie.Title)
    }
    log.Println(strings.Repeat("-", 88))

    log.Printf("Getting data again to verify the update.")
    movie, err = runner.GetMovie(customMovie.Title, customMovie.Year)
    if err == nil {
        log.Println(movie)
    }
    log.Println(strings.Repeat("-", 88))

    log.Printf("Deleting movie '%v'.\n", customMovie.Title)
    err = runner.DeleteMovie(customMovie)
    if err == nil {
        log.Printf("Deleted %v.\n", customMovie.Title)
    }

    err = tableBasics.DeleteTable()
    if err == nil {
        log.Printf("Deleted table %v.\n", tableBasics.TableName)
    }

    log.Println(strings.Repeat("-", 88))
    log.Println("Thanks for watching!")
    log.Println(strings.Repeat("-", 88))
}

```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Go API Reference*.

## IAM examples using SDK for Go V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Go V2 with AWS Identity and Access Management.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2746\)](#)
- [Scenarios \(p. 2767\)](#)

## Actions

### Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

#### SDK for Go V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// AttachRolePolicy

_, err = service.AttachRolePolicy(context.Background(), &iam.AttachRolePolicyInput{
    PolicyArn: aws.String(ExamplePolicyARN),
    RoleName:  aws.String(ExampleRoleName),
})

if err != nil {
    panic("Couldn't apply a policy to the role!")
}

fmt.Println("## Attached policy " + ExamplePolicyARN + " to role " + ExampleRoleName)
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Go API Reference*.

### Attach a policy to a user

The following code example shows how to attach an IAM policy to a user.

#### SDK for Go V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
```

```

"fmt"

"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMAttachRolePolicyAPI defines the interface for the AttachRolePolicy function.
// We use this interface to test the function using a mocked service.
type IAMAttachRolePolicyAPI interface {
    AttachRolePolicy(ctx context.Context,
        params *iam.AttachRolePolicyInput,
        optFns ...func(*iam.Options)) (*iam.AttachRolePolicyOutput, error)
}

// AttachDynamoFullPolicy attaches an Amazon DynamoDB full-access policy to an AWS
// Identity and Access Management (IAM) role.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, an AttachRolePolicyOutput object containing the result of the
//     service call and nil.
//     Otherwise, nil and an error from the call to AttachRolePolicy.
func AttachDynamoFullPolicy(c context.Context, api IAMAttachRolePolicyAPI, input
    *iam.AttachRolePolicyInput) (*iam.AttachRolePolicyOutput, error) {
    return api.AttachRolePolicy(c, input)
}

func main() {
    roleName := flag.String("r", "", "The name of the IAM role")
    policyName := flag.String("p", "", "The name of the policy to attach to the role")
    flag.Parse()

    if *roleName == "" || *policyName == "" {
        fmt.Println("You must supply a role and policy name (-r ROLE -p POLICY)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    policyArn := "arn:aws:iam::aws:policy/" + *policyName

    input := &iam.AttachRolePolicyInput{
        PolicyArn: &policyArn,
        RoleName:  roleName,
    }

    _, err = AttachDynamoFullPolicy(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Unable to attach policy " + *policyName + " to role " + *roleName)
        return
    }

    fmt.Println("Policy " + *policyName + " attached to role " + *roleName)
}

```

- For API details, see [AttachUserPolicy](#) in *AWS SDK for Go API Reference*.

## Create a policy

The following code example shows how to create an IAM policy.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// CreatePolicy

fmt.Println("# CreatePolicy")

policyDocument := `{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb>DeleteItem",
                "dynamodb>GetItem",
                "dynamodb>PutItem",
                "dynamodb>Query",
                "dynamodb>Scan",
                "dynamodb>UpdateItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-2:123456789012:table/mytable",
                "arn:aws:dynamodb:us-west-2:123456789012:table/mytable/*"
            ]
        }
    ]
}`

createPolicyResult, err := service.CreatePolicy(context.Background(),
&iam.CreatePolicyInput{
    PolicyDocument: &policyDocument,
    PolicyName:     aws.String(ExamplePolicyName),
})

if err != nil {
    panic("Couldn't create policy!" + err.Error())
}

fmt.Print("Created a new policy: " + *createPolicyResult.Policy.Arn)
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Go API Reference*.

## Create a role

The following code example shows how to create an IAM role.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// CreateRole
```

```
myRole, err := service.CreateRole(context.Background(), &iam.CreateRoleInput{
    RoleName: aws.String(ExampleRoleName),
    Description: aws.String("My super awesome example role"),
    AssumeRolePolicyDocument: aws.String(`{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "ec2.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }`),
})
if err != nil {
    panic("Couldn't create role: " + err.Error())
}
fmt.Println("## Create Role")
fmt.Printf("The new role's ARN is %s \n", *myRole.Role.Arn)
```

- For API details, see [CreateRole](#) in *AWS SDK for Go API Reference*.

## Create a service-linked role

The following code example shows how to create an IAM service-linked role.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// CreateServiceLinkedRole

fmt.Println("## Create SLR for " + ExampleSLRService)
createSlrResult, err := service.CreateServiceLinkedRole(context.Background(),
&iam.CreateServiceLinkedRoleInput{
    AWSServiceName: aws.String(ExampleSLRService),
    Description:   aws.String(ExampleSLRDescription),
})

// NOTE: We don't consider this an error as running this example multiple times will
// cause an error.
if err != nil {
    fmt.Printf("Couldn't create service-linked role: %v\n", err.Error())
} else {

    fmt.Printf("Created service-linked role with ARN: %s\n", *createSlrResult.Role.Arn)
}
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for Go API Reference*.

## Create a user

The following code example shows how to create an IAM user.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// CreateUser

fmt.Println("## Create user " + ExampleUserName)

createUserResult, err := service.CreateUser(context.Background(),
&iam.CreateUserInput{
    UserName: aws.String(ExampleUserName),
})

if err != nil {
    panic("Couldn't create user: " + err.Error())
}

fmt.Printf("Created user %s\n", *createUserResult.User.Arn)
```

- For API details, see [CreateUser](#) in *AWS SDK for Go API Reference*.

## Create an access key

The following code example shows how to create an IAM access key.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
)

// IMACreateAccessKeyAPI defines the interface for the CreateAccessKey function.
// We use this interface to test the function using a mocked service.
type IAMCreateAccessKeyAPI interface {
    CreateAccessKey(ctx context.Context,
        params *iam.CreateAccessKeyInput,
        optFns ...func(*iam.Options)) (*iam.CreateAccessKeyOutput, error)
}

// MakeAccessKey creates a new AWS Identity and Access Management (IAM) access key for
// a user.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
```

```
//      api is the interface that defines the method call.
//      input defines the input arguments to the service call.
// Output:
//      If successful, a CreateAccessKeyOutput object containing the result of the
//      service call and nil.
//      Otherwise, nil and an error from the call to CreateAccessKey.
func MakeAccessKey(c context.Context, api IAMCreateAccessKeyAPI, input
    *iam.CreateAccessKeyInput) (*iam.CreateAccessKeyOutput, error) {
    return api.CreateAccessKey(c, input)
}

func main() {
    userName := flag.String("u", "", "The name of the user")
    flag.Parse()

    if *userName == "" {
        fmt.Println("You must supply a user name (-u USER)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.CreateAccessKeyInput{
        UserName: userName,
    }

    result, err := MakeAccessKey(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error creating a new access key")
        fmt.Println(err)
        return
    }

    fmt.Println("Created new access key with ID: " + *result.AccessKeyId + " and
    secret key: " + *result.AccessKey.SecretAccessKey)
}
```

- For API details, see [CreateAccessKey in AWS SDK for Go API Reference](#).

## Create an alias for an account

The following code example shows how to create an alias for an IAM account.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"
```

```
"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMCreateAccountAliasAPI defines the interface for the CreateAccountAlias function.
// We use this interface to test the function using a mocked service.
type IAMCreateAccountAliasAPI interface {
    CreateAccountAlias(ctx context.Context,
        params *iam.CreateAccountAliasInput,
        optFns ...func(*iam.Options)) (*iam.CreateAccountAliasOutput, error)
}

// MakeAccountAlias creates an alias for your AWS Identity and Access Management (IAM)
// account.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, a CreateAccountAliasOutput object containing the result of the
//     service call and nil.
//     Otherwise, nil and an error from the call to CreateAccountAlias.
func MakeAccountAlias(c context.Context, api IAMCreateAccountAliasAPI, input
    *iam.CreateAccountAliasInput) (*iam.CreateAccountAliasOutput, error) {
    return api.CreateAccountAlias(c, input)
}

func main() {
    alias := flag.String("a", "", "The account alias")
    flag.Parse()

    if *alias == "" {
        fmt.Println("You must supply an account alias (-a ALIAS)")
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.CreateAccountAliasInput{
        AccountAlias: alias,
    }

    _, err = MakeAccountAlias(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error creating an account alias")
        fmt.Println(err)
        return
    }

    fmt.Printf("Created account alias " + *alias)
}
```

- For API details, see [CreateAccountAlias](#) in [AWS SDK for Go API Reference](#).

## Delete a user

The following code example shows how to delete an IAM user.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// DeleteUser

_, err = service.DeleteUser(context.Background(), &iam.DeleteUserInput{
    UserName: aws.String(ExampleUserName),
})

if err != nil {
    panic("Couldn't delete user: " + err.Error())
}
```

- For API details, see [DeleteUser](#) in *AWS SDK for Go API Reference*.

## Delete an access key

The following code example shows how to delete an IAM access key.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMDeleteAccessKeyAPI defines the interface for the DeleteAccessKey function.
// We use this interface to test the function using a mocked service.
type IAMDeleteAccessKeyAPI interface {
    DeleteAccessKey(ctx context.Context,
        params *iam.DeleteAccessKeyInput,
        optFns ...func(*iam.Options)) (*iam.DeleteAccessKeyOutput, error)
}

// RemoveAccessKey deletes an AWS Identity and Access Management (IAM) access key.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, a DeleteAccessKeyOutput object containing the result of the
//     service call and nil.
//     Otherwise, nil and an error from the call to DeleteAccessKey.
func RemoveAccessKey(c context.Context, api IAMDeleteAccessKeyAPI, input
    *iam.DeleteAccessKeyInput) (*iam.DeleteAccessKeyOutput, error) {
    return api.DeleteAccessKey(c, input)
}
```

```
func main() {
    keyID := flag.String("k", "", "The ID of the access key")
    userName := flag.String("u", "", "The name of the user")
    flag.Parse()

    if *keyID == "" || *userName == "" {
        fmt.Println("You must supply the key ID and user name (-k KEY-ID -u USER-NAME")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.DeleteAccessKeyInput{
        AccessKeyId: keyID,
        UserName:    userName,
    }

    _, err = RemoveAccessKey(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Error", err)
        return
    }

    fmt.Println("Deleted key with ID " + *keyID + " from user " + *userName)
}
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for Go API Reference*.

## Detach a policy from a user

The following code example shows how to detach an IAM policy from a user.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMDetachRolePolicyAPI defines the interface for the DetachRolePolicy function.
// We use this interface to test the function using a mocked service.
type IAMDetachRolePolicyAPI interface {
    DetachRolePolicy(ctx context.Context,
        params *iam.DetachRolePolicyInput,
```

```
    optFns ...func(*iam.Options)) (*iam.DetachRolePolicyOutput, error)
}

// DetachDynamoFullPolicy detaches an Amazon DynamoDB full-access policy from an AWS
// Identity and Access Management (IAM) role.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If successful, a DetachRolePolicyOutput object containing the result of the
//   service call and nil.
//   Otherwise, nil and an error from the call to DetachRolePolicy.
func DetachDynamoFullPolicy(c context.Context, api IAMDetachRolePolicyAPI, input
    *iam.DetachRolePolicyInput) (*iam.DetachRolePolicyOutput, error) {
    return api.DetachRolePolicy(c, input)
}

func main() {
    roleName := flag.String("r", "", "The name of the IAM role")
    flag.Parse()

    if *roleName == "" {
        fmt.Println("You must supply a role name (-r ROLE)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    policyArn := "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess"
    input := &iam.DetachRolePolicyInput{
        PolicyArn: &policyArn,
        RoleName:  roleName,
    }

    _, err = DetachDynamoFullPolicy(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Unable to detach DynamoDB full-access role policy from role")
        return
    }
    fmt.Println("Role detached successfully")
}
```

- For API details, see [DetachUserPolicy](#) in [AWS SDK for Go API Reference](#).

## Get a policy

The following code example shows how to get an IAM policy.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// GetPolicy
```

```
getPolicyResponse, err := service.GetPolicy(context.Background(), &iam.GetPolicyInput{  
    PolicyArn: policyArn,  
})  
  
if err != nil {  
    panic("Couldn't get policy from ARN: " + err.Error())  
}  
  
fmt.Printf("policy: %s, name %s\n",  
    *getPolicyResponse.Policy.Arn,  
    *getPolicyResponse.Policy.PolicyName)
```

- For API details, see [GetPolicy](#) in *AWS SDK for Go API Reference*.

## Get a role

The following code example shows how to get an IAM role.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// GetRole  
  
getRoleResult, err := service.GetRole(context.Background(), &iam.GetRoleInput{  
    RoleName: aws.String(ExampleRoleName),  
})  
  
if err != nil {  
    panic("Couldn't get role! " + err.Error())  
}  
  
fmt.Println("## GetRole results: ")  
fmt.Println("ARN: ", *getRoleResult.Role.Arn)  
fmt.Println("Name: ", *getRoleResult.Role.RoleName)  
fmt.Println("Created On: ", *getRoleResult.Role.CreateDate)
```

- For API details, see [GetRole](#) in *AWS SDK for Go API Reference*.

## Get data about the last use of an access key

The following code example shows how to get data about the last use of an IAM access key.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main  
  
import (
```

```
"context"
"flag"
"fmt"

"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMGetAccessKeyLastUsedAPI defines the interface for the GetAccessKeyLastUsed
// function.
// We use this interface to test the function using a mocked service.
type IAMGetAccessKeyLastUsedAPI interface {
    GetAccessKeyLastUsed(ctx context.Context,
        params *iam.GetAccessKeyLastUsedInput,
        optFns ...func(*iam.Options)) (*iam.GetAccessKeyLastUsedOutput, error)
}

// WhenWasKeyUsed retrieves when an AWS Identity and Access Management (IAM) access key
// was last used, including the AWS Region and with which service.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, a GetAccessKeyLastUsedOutput object containing the result of the
//     service call and nil.
//     Otherwise, nil and an error from the call to GetAccessKeyLastUsed.
func WhenWasKeyUsed(c context.Context, api IAMGetAccessKeyLastUsedAPI, input
    *iam.GetAccessKeyLastUsedInput) (*iam.GetAccessKeyLastUsedOutput, error) {
    return api.GetAccessKeyLastUsed(c, input)
}

func main() {
    keyID := flag.String("k", "", "The ID of the access key")
    flag.Parse()

    if *keyID == "" {
        fmt.Println("You must supply the ID of an access key (-k KEY-ID)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.GetAccessKeyLastUsedInput{
        AccessKeyId: keyID,
    }

    result, err := WhenWasKeyUsed(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error retrieving when access key was last used:")
        fmt.Println(err)
        return
    }

    fmt.Println("The key was last used:", *result.AccessKeyLastUsed)
}
```

- For API details, see [GetAccessKeyLastUsed](#) in *AWS SDK for Go API Reference*.

## Get the account password policy

The following code example shows how to get the IAM account password policy.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// GetAccountPasswordPolicy

fmt.Println("# GetAccountPasswordPolicy")

accountPasswordPolicy, err := service.GetAccountPasswordPolicy(context.Background(),
    &iam.GetAccountPasswordPolicyInput{})

if err != nil {
    var notexists *types.NoSuchEntityException
    if errors.As(err, &notexists) {
        fmt.Println("No password policy")
    } else {
        panic("Couldn't get account password policy! " + err.Error())
    }
} else {
    fmt.Println("Users can change password: ",
        accountPasswordPolicy.PasswordPolicy.AllowUsersToChangePassword)
    fmt.Println("Passwords expire: ",
        accountPasswordPolicy.PasswordPolicy.ExpirePasswords)
    fmt.Println("Minimum password length: ",
        accountPasswordPolicy.PasswordPolicy.MinimumPasswordLength)
}
```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for Go API Reference*.

## List SAML providers

The following code example shows how to list SAML providers for IAM.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// ListSAMLProviders

samlProviderList, err := service.ListSAMLProviders(context.Background(),
    &iam.ListSAMLProvidersInput{})

if err != nil {
    panic("Couldn't list saml providers: " + err.Error())
}

for _, provider := range samlProviderList.SAMLProviderList {
    fmt.Printf("%s %s -> %s", *provider.Arn, *provider.CreateDate, *provider.ValidUntil)
}
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for Go API Reference*.

## List a user's access keys

The following code example shows how to list a user's IAM access keys.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMListAccessKeysAPI defines the interface for the ListAccessKeys function.
// We use this interface to test the function using a mocked service.
type IAMListAccessKeysAPI interface {
    ListAccessKeys(ctx context.Context,
        params *iam.ListAccessKeysInput,
        optFns ...func(*iam.Options)) (*iam.ListAccessKeysOutput, error)
}

// GetAccessKeys retrieves up to the AWS Identity and Access Management (IAM) access
// keys for a user.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, a ListAccessKeysOutput object containing the result of the
//     service call and nil.
//     Otherwise, nil and an error from the call to ListAccessKeys.
func GetAccessKeys(c context.Context, api IAMListAccessKeysAPI, input
    *iam.ListAccessKeysInput) (*iam.ListAccessKeysOutput, error) {
    return api.ListAccessKeys(c, input)
}

func main() {
    maxItems := flag.Int("m", 10, "The maximum number of access keys to show")
    userName := flag.String("u", "", "The name of the user")
    flag.Parse()

    if *userName == "" {
        fmt.Println("You must supply the name of a user (-u USER)")
        return
    }

    if *maxItems < 0 {
        *maxItems = 10
    }
}
```

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic("configuration error, " + err.Error())
}

client := iam.NewFromConfig(cfg)

input := &iam.ListAccessKeysInput{
    MaxItems: aws.Int32(int32(*maxItems)),
    UserName: userName,
}

result, err := GetAccessKeys(context.TODO(), client, input)
if err != nil {
    fmt.Println("Got an error retrieving user access keys:")
    fmt.Println(err)
    return
}

for _, key := range result.AccessKeyMetadata {
    fmt.Println("Status for access key " + *key.AccessKeyId + ": " + string(key.Status))
}
}
```

- For API details, see [ListAccessKeys](#) in *AWS SDK for Go API Reference*.

## List account aliases

The following code example shows how to list IAM account aliases.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMListAccountAliasesAPI defines the interface for the ListAccountAliases function.
// We use this interface to test the function using a mocked service.
type IAMListAccountAliasesAPI interface {
    ListAccountAliases(ctx context.Context,
        params *iam.ListAccountAliasesInput,
        optFns ...func(*iam.Options)) (*iam.ListAccountAliasesOutput, error)
}

// GetAccountAliases retrieves the aliases for your AWS Identity and Access Management
// (IAM) account.
// Inputs:
```

```
//      c is the context of the method call, which includes the AWS Region.
//      api is the interface that defines the method call.
//      input defines the input arguments to the service call.
// Output:
//      If successful, a ListAccountAliasesOutput object containing the result of the
//      service call and nil.
//      Otherwise, nil and an error from the call to ListAccountAliases.
func GetAccountAliases(c context.Context, api IAMListAccountAliasesAPI, input
    *iam.ListAccountAliasesInput) (*iam.ListAccountAliasesOutput, error) {
    return api.ListAccountAliases(c, input)
}

func main() {
    maxItems := flag.Int("m", 10, "Maximum number of aliases to list")
    flag.Parse()

    if *maxItems < 0 {
        *maxItems = 10
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.ListAccountAliasesInput{
        MaxItems: aws.Int32(int32(*maxItems)),
    }

    result, err := GetAccountAliases(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error retrieving account aliases")
        fmt.Println(err)
        return
    }

    for i, alias := range result.AccountAliases {
        fmt.Printf("Alias %d: %s\n", i, alias)
    }
}
```

- For API details, see [ListAccountAliases](#) in *AWS SDK for Go API Reference*.

## List groups

The following code example shows how to list IAM groups.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// ListGroups

listGroupsResult, err := service.ListGroups(context.Background(),
    &iam.ListGroupsInput{})
```

```
if err != nil {
    panic("Couldn't list groups! " + err.Error())
}

for _, group := range listGroupsResult.Groups {
    fmt.Printf("group %s - %s\n", *group.GroupId, *group.Arn)
}
```

- For API details, see [ListGroups](#) in *AWS SDK for Go API Reference*.

### List inline policies for a role

The following code example shows how to list inline policies for an IAM role.

#### SDK for Go V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// ListRolePolicies

rolePoliciesList, err := service.ListRolePolicies(context.Background(),
&iam.ListRolePoliciesInput{
    RoleName: aws.String(ExampleRoleName),
})

if err != nil {
    panic("Couldn't list policies for role: " + err.Error())
}

for _, rolePolicy := range rolePoliciesList.PolicyNames {
    fmt.Printf("Policy ARN: %v", rolePolicy)
}
```

- For API details, see [ListRolePolicies](#) in *AWS SDK for Go API Reference*.

### List policies

The following code example shows how to list IAM policies.

#### SDK for Go V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// ListPolicies

policyListResponse, err := service.ListPolicies(context.Background(),
&iam.ListPoliciesInput{})

if err != nil {
    panic("Couldn't get list of policies! " + err.Error())
}
```

```
fmt.Println("PolicyName\tARN")
for _, policy := range policyListResponse.Policies {
    fmt.Printf("%s\t%s\n", *policy.PolicyName, *policy.Arn)
}
```

- For API details, see [ListPolicies](#) in *AWS SDK for Go API Reference*.

### List policies attached to a role

The following code example shows how to list policies attached to an IAM role.

#### SDK for Go V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// ListAttachedRolePolicies

attachedPoliciesList, err := service.ListAttachedRolePolicies(context.Background(),
    &iam.ListAttachedRolePoliciesInput{
        RoleName: aws.String(ExampleRoleName),
    })

if err != nil {
    panic("Couldn't call ListAttachedRolePolicies: " + err.Error())
}

fmt.Println("## List attached role policies for " + ExampleRoleName)

for _, attachedPolicy := range attachedPoliciesList.AttachedPolicies {
    fmt.Printf("attached policy: %v\n (%v) \n", attachedPolicy.PolicyArn,
        attachedPolicy.PolicyName)
}
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for Go API Reference*.

### List roles

The following code example shows how to list IAM roles.

#### SDK for Go V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// ListRoles

roles, err := service.ListRoles(context.Background(), &iam.ListRolesInput{})

if err != nil {
    panic("Could not list roles: " + err.Error())
}
```

```
fmt.Println("## list roles")
for _, idxRole := range roles.Roles {
    fmt.Printf("%s\t%s\t%s\t",
        *idxRole.RoleId,
        *idxRole.RoleName,
        *idxRole.Arn)
    if idxRole.Description != nil {
        fmt.Print(*idxRole.Description)
    }
    fmt.Println("\n")
}
```

- For API details, see [ListRoles](#) in *AWS SDK for Go API Reference*.

## List users

The following code example shows how to list IAM users.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// ListUsers

fmt.Println("## List users")

userListResult, err := service.ListUsers(context.Background(), &iam.ListUsersInput{})
if err != nil {
    panic("Couldn't list users: " + err.Error())
}
for _, userResult := range userListResult.Users {
    fmt.Printf("%s\t%s\n", *userResult.UserName, *userResult.Arn)
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Go API Reference*.

## Update a user

The following code example shows how to update an IAM user.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"
```

```
"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/iam"
)

// IAMUpdateUserAPI defines the interface for the UpdateUser function.
// We use this interface to test the function using a mocked service.
type IAMUpdateUserAPI interface {
    UpdateUser(ctx context.Context,
        params *iam.UpdateUserInput,
        optFns ...func(*iam.Options)) (*iam.UpdateUserOutput, error)
}

// RenameUser changes the name for an AWS Identity and Access Management (IAM) user.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, a UpdateUserOutput object containing the result of the service
//     call and nil.
//     Otherwise, nil and an error from the call to UpdateUser.
func RenameUser(c context.Context, api IAMUpdateUserAPI, input *iam.UpdateUserInput)
    (*iam.UpdateUserOutput, error) {
    return api.UpdateUser(c, input)
}

func main() {
    userName := flag.String("u", "", "The name of the user")
    newName := flag.String("n", "", "The new name of the user")
    flag.Parse()

    if *userName == "" || *newName == "" {
        fmt.Println("You must supply a user name and new name (-u USERNAME -n NEW-NAME)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.UpdateUserInput{
        UserName:    userName,
        NewUserName: newName,
    }

    _, err = RenameUser(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error updating user " + *userName)
    }

    fmt.Println("Updated user name from: " + *userName + " to: " + *newName)
}
```

- For API details, see [UpdateUser](#) in [AWS SDK for Go API Reference](#).

## Update an access key

The following code example shows how to update an IAM access key.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// IAMUpdateAccessKeyAPI defines the interface for the UpdateAccessKey function.
// We use this interface to test the function using a mocked service.
type IAMUpdateAccessKeyAPI interface {
    UpdateAccessKey(ctx context.Context,
        params *iam.UpdateAccessKeyInput,
        optFns ...func(*iam.Options)) (*iam.UpdateAccessKeyOutput, error)
}

// ActivateKey sets the status of an AWS Identity and Access Management (IAM) access
// key to active.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If successful, a UpdateAccessKeyOutput object containing the result of the
//   service call and nil.
//   Otherwise, nil and an error from the call to UpdateAccessKey.
func ActivateKey(c context.Context, api IAMUpdateAccessKeyAPI, input
    *iam.UpdateAccessKeyInput) (*iam.UpdateAccessKeyOutput, error) {
    return api.UpdateAccessKey(c, input)
}

func main() {
    keyID := flag.String("k", "", "The ID of the access key")
    userName := flag.String("u", "", "The name of the user")
    flag.Parse()

    if *keyID == "" || *userName == "" {
        fmt.Println("You must supply an access key ID and user name (-k KEY-ID -u USER-NAME)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.UpdateAccessKeyInput{
        AccessKeyId: keyID,
        Status:      types.StatusTypeActive,
        UserName:    userName,
    }
}
```

```
_ , err = ActivateKey(context.TODO(), client, input)
if err != nil {
    fmt.Println("Error", err)
    return
}

fmt.Println("Access Key activated")
```

- For API details, see [UpdateAccessKey](#) in *AWS SDK for Go API Reference*.

## Scenarios

### Create a user and assume a role

The following code example shows how to:

- Create a user who has no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list Amazon S3 buckets using temporary credentials.
- Delete the policy, role, and user.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "errors"
    "fmt"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/credentials"
    "github.com/aws/aws-sdk-go-v2/credentials/stscreds"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/sts"
    "github.com/aws/smithy-go"
)

/*
This is a basic scenario for using AWS Identity and Access Management (IAM) and AWS
Security Token Service (STS).

This example will do the following:
* Create a user that has no permissions.
* Create a role and policy that grant s3>ListAllMyBuckets permission.

```

```
*   Grant the user permission to assume the role.
*   Create an S3 client object as the user and try to list buckets (this should fail).
*   Get temporary credentials by assuming the role.
*   Create an S3 client object with the temporary credentials and list the buckets
(this should succeed).
*   Delete all the resources.

*/
const (
    scenario_UserName          = "ExampleUser123"
    scenario_BucketListerRoleName = "BucketListerRole"
    scenario_BucketListerPolicyName = "BucketListerListMyBucketsPolicy"
)

func scenario() {
    cfg, err := config.LoadDefaultConfig(context.Background())
    if err != nil {
        panic("Couldn't load a configuration")
    }

    iamSvc := iam.NewFromConfig(cfg)
    fmt.Println("## Create user")

    var userInfo types.User

    user, err := iamSvc.CreateUser(context.Background(), &iam.CreateUserInput{
        UserName: aws.String(scenario_UserName),
    })
    if err != nil {
        var existsAlready *types.EntityAlreadyExistsException
        if errors.As(err, &existsAlready) {
            // The user possibly exists.
            // Check if the user actually exists within IAM.
            user, err := iamSvc.GetUser(context.Background(), &iam.GetUserInput{UserName:
aws.String(scenario_UserName)})
            if err != nil {
                panic("Can't get user: " + err.Error())
            } else {
                // Make sure the user info is set for later.
                userInfo = *user.User
                fmt.Println("User already existed...")
            }
        } else {
            fmt.Println("Couldn't create user! " + err.Error())
        }
    } else {
        userInfo = *user.User
    }

    fmt.Printf("User %s has id %s\n", scenario_UserName, *userInfo.Arn)
    fmt.Println("## Create access key")

    creds, err := iamSvc.CreateAccessKey(context.Background(), &iam.CreateAccessKeyInput{
        UserName: aws.String(scenario_UserName),
    })
    if err != nil {
        panic("Couldn't create credentials for a user! " + err.Error())
    }
}
```

```
akId := *creds.AccessKey.AccessKeyId
sakId := *creds.AccessKey.SecretAccessKey

fmt.Printf("## CREDs: accessKeyId(%s) Secretkey(%s)\n", akId, sakId)

// Grant the user the ability to assume the role.

fmt.Println("# waiting for a few moments for keys to become available")

time.Sleep(10 * time.Second)

fmt.Println("## Create the bucket lister role")
bucketListerRole, err := iamSvc.CreateRole(context.Background(), &iam.CreateRoleInput{
    RoleName: aws.String(scenario_BucketListerRoleName),
    AssumeRolePolicyDocument: aws.String(`{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "AWS": ` + (*userInfo.Arn) + `"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    `),
    Description: aws.String("Role to let users list their buckets."),
})
var bucketListerRoleArn string
if err != nil {
    // Check to see if the role exists.
    var existsException *types.EntityAlreadyExistsException
    if errors.As(err, &existsException) {
        // Check if we can look up the role as it stands already
        tRole, err := iamSvc.GetRole(context.Background(), &iam.GetRoleInput{
            RoleName: aws.String(scenario_BucketListerRoleName),
        })
        if err != nil {
            // Told it already exists, but now it's gone.
            panic("Couldn't find seemingly extant role: " + err.Error())
        } else {
            bucketListerRoleArn = *tRole.Role.Arn
        }
    } else {
        panic("Couldn't create role! " + err.Error())
    }
} else {
    bucketListerRoleArn = *bucketListerRole.Role.Arn
}

fmt.Printf("## The ARN for the bucket lister role is %s", bucketListerRoleArn)

fmt.Println("## Create policy to allow bucket listing")
bucketAllowPolicy := aws.String(`{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1646154730759",
            "Action": [
                "s3>ListAllMyBuckets"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}`)

var bucketListerPolicyArn string
```

```
bucketListerPolicy, err := iamSvc.CreatePolicy(context.Background(),
&iam.CreatePolicyInput{
    PolicyName: aws.String(scenario_BucketListerPolicyName),
    PolicyDocument: bucketAllowPolicy,
    Description: aws.String("Allow user to list their own buckets"),
})

if err != nil {
    var existsException *types.EntityAlreadyExistsException
    if errors.As(err, &existsException) {

        stsClient := sts.NewFromConfig(cfg)
        mCallerId, _ := stsClient.GetCallerIdentity(context.Background(),
&sts.GetCallerIdentityInput{})

        mpolicyArn := fmt.Sprintf("arn:aws:iam::%s:policy/%s", *mCallerId.Account,
scenario_BucketListerPolicyName)

        _, err := iamSvc.GetPolicy(context.Background(), &iam.GetPolicyInput{
            PolicyArn: &mpolicyArn,
        })
        if err != nil {
            panic("Failed to find policy by arn(" + mpolicyArn + ") -> " + err.Error())
        }
        bucketListerPolicyArn = mpolicyArn

    } else {
        panic("Couldn't create policy! " + err.Error())
    }
} else {
    bucketListerPolicyArn = *bucketListerPolicy.Policy.Arn
}

fmt.Println("## Attach role policy")

_, err = iamSvc.AttachRolePolicy(context.Background(), &iam.AttachRolePolicyInput{
    PolicyArn: &bucketListerPolicyArn,
    RoleName: aws.String(scenario_BucketListerRoleName),
})

if err != nil {
    fmt.Println("Couldn't attach policy to role! " + err.Error())
}

fmt.Printf("# attached role policy ")

fmt.Println("# waiting for a few moments for keys to become available")

time.Sleep(10 * time.Second)

// Create an S3 client that acts as the user.
userConfig, err := config.LoadDefaultConfig(context.TODO(),
    // Use credentials created earlier.
    config.WithCredentialsProvider(credentials.StaticCredentialsProvider{
        Value: aws.Credentials{
            AccessKeyID: akId, SecretAccessKey: sakId,
            Source: "Example user creds",
        },
    }))
if err != nil {
    panic("Couldn't create config for new credentials! " + err.Error())
} else {
    creds, err := userConfig.Credentials.Retrieve(context.Background())
    if err != nil {
        panic("Couldn't get credentials for our new config, something is wrong: " +
err.Error())
    }
}
```

```

        }
        fmt.Printf("-> config creds are %s %s\n", creds.AccessKeyID, creds.SecretAccessKey)
    }

s3Client := s3.NewFromConfig(userConfig)
// Attempt to list buckets.
_, err = s3Client.ListBuckets(context.Background(), &s3.ListBucketsInput{})

if err == nil {
    fmt.Println("Call to s3>ListBuckets was not denied (unexpected!)")
} else {
    var oe smithy.APIError
    if errors.As(err, &oe) && (oe.ErrorCode() == "AccessDenied") {
        fmt.Println("Couldn't list buckets (expected!)")
    } else {
        panic("unexpected error: " + err.Error())
    }
}

stsClient := sts.NewFromConfig(userConfig)
rolecreds := stscreds.NewAssumeRoleProvider(stsClient, bucketListerRoleArn)

tmpCreds, err := rolecreds.Retrieve(context.Background())
if err != nil {
    fmt.Println("couldn't get role creds: " + err.Error())
} else {
    fmt.Printf("role creds: %s %v\n\n", tmpCreds.AccessKeyID, tmpCreds.Expires)
}

roleConfig, err := config.LoadDefaultConfig(context.Background(),
    config.WithCredentialsProvider(rolecreds),
)
if err != nil {
    panic("Couldn't create config with assumed role! " + err.Error())
}

roleS3client := s3.NewFromConfig(roleConfig)

mybuckets, err := roleS3client.ListBuckets(context.Background(),
    &s3.ListBucketsInput{}
)

if err != nil {
    panic("Couldn't list buckets while assuming role that allows this: " + err.Error())
} else {
    fmt.Println("Buckets owned by the user....")
    for _, bucket := range mybuckets.Buckets {
        fmt.Printf("%s -> %s\n", *bucket.Name, bucket.CreationDate)
    }
}

// ---- Clean up ----

// Delete the user's access keys.

fmt.Println("cleanup: Delete created access key")
_, _ = iamSvc.DeleteAccessKey(context.Background(), &iam.DeleteAccessKeyInput{
    AccessKeyId: &akId,
    UserName: aws.String(scenario_UserName),
})

// Delete the user.
fmt.Println("cleanup: delete the user we created")
_, err = iamSvc.DeleteUser(context.Background(), &iam.DeleteUserInput{UserName:
aws.String(scenario_UserName)})
if err != nil {
    fmt.Println("Couldn't delete user! " + err.Error())
}

```

```
}

// Detach the role policy.

fmt.Println("cleanup: Detach the policy from the role")
_, err = iamSvc.DetachRolePolicy(context.Background(), &iam.DetachRolePolicyInput{
    RoleName: aws.String(scenario_BucketListerRoleName),
    PolicyArn: &bucketListerPolicyArn,
})

if err != nil {
    fmt.Println("Couldn't detach role policy from role " + err.Error())
}

// Delete the role.

fmt.Println("cleanup: Remove the role")
_, err = iamSvc.DeleteRole(context.Background(), &iam.DeleteRoleInput{
    RoleName: aws.String(scenario_BucketListerRoleName),
})
if err != nil {
    fmt.Println("Couldn't delete role! " + err.Error())
}

// Delete the policy.

fmt.Println("cleanup: delete the policy")
_, err = iamSvc.DeletePolicy(context.Background(), &iam.DeletePolicyInput{
    PolicyArn: &bucketListerPolicyArn,
})
if err != nil {
    fmt.Println("couldn't delete policy!")
}

fmt.Println("done!")
```

- For API details, see the following topics in *AWS SDK for Go API Reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## AWS KMS examples using SDK for Go V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Go V2 with AWS Key Management Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2773\)](#)

## Actions

### Create a key

The following code example shows how to create an AWS KMS key.

#### SDK for Go V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/kms"
)

// KMSCreateKeyAPI defines the interface for the CreateKey function.
// We use this interface to test the function using a mocked service.
type KMSCreateKeyAPI interface {
    CreateKey(ctx context.Context,
        params *kms.CreateKeyInput,
        optFns ...func(*kms.Options)) (*kms.CreateKeyOutput, error)
}

// MakeKey creates an AWS Key Management Service (AWS KMS) key (KMS key).
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If success, a CreateKeyOutput object containing the result of the service call
//     and nil.
//     Otherwise, nil and an error from the call to CreateKey.
func MakeKey(c context.Context, api KMSCreateKeyAPI, input *kms.CreateKeyInput)
    (*kms.CreateKeyOutput, error) {
    return api.CreateKey(c, input)
}

func main() {
    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := kms.NewFromConfig(cfg)

    input := &kms.CreateKeyInput{}

    result, err := MakeKey(context.TODO(), client, input)
    if err != nil {
```

```
    fmt.Println("Got error creating key:")
    fmt.Println(err)
    return
}

fmt.Println(*result.KeyMetadata.KeyId)
}
```

- For API details, see [CreateKey](#) in *AWS SDK for Go API Reference*.

## Decrypt ciphertext

The following code example shows how to decrypt ciphertext that was encrypted by a KMS key.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    b64 "encoding/base64"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/kms"
)

// KMSDecryptAPI defines the interface for the Decrypt function.
// We use this interface to test the function using a mocked service.
type KMSDecryptAPI interface {
    Decrypt(ctx context.Context,
        params *kms.DecryptInput,
        optFns ...func(*kms.Options)) (*kms.DecryptOutput, error)
}

// DecodeData decrypts some text that was encrypted with an AWS Key Management Service
// (AWS KMS) key (KMS key).
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If success, a DecryptOutput object containing the result of the service call and
//     nil.
//     Otherwise, nil and an error from the call to Decrypt.
func DecodeData(c context.Context, api KMSDecryptAPI, input *kms.DecryptInput)
    (*kms.DecryptOutput, error) {
    return api.Decrypt(c, input)
}

func main() {
    data := flag.String("d", "", "The encrypted data, as a string")
    flag.Parse()

    if *data == "" {
```

```
    fmt.Println("You must supply the encrypted data as a string")
    fmt.Println("-d DATA")
    return
}

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic("configuration error, " + err.Error())
}

client := kms.NewFromConfig(cfg)

blob, err := b64.StdEncoding.DecodeString(*data)
if err != nil {
    panic("error converting string to blob, " + err.Error())
}

input := &kms.DecryptInput{
    CiphertextBlob: blob,
}

result, err := DecodeData(context.TODO(), client, input)
if err != nil {
    fmt.Println("Got error decrypting data: ", err)
    return
}

fmt.Println(string(result.Plaintext))
}
```

- For API details, see [Decrypt](#) in *AWS SDK for Go API Reference*.

## Encrypt text using a key

The following code example shows how to encrypt text using a KMS key.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    b64 "encoding/base64"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/kms"
)

// KMSEncryptAPI defines the interface for the Encrypt function.
// We use this interface to test the function using a mocked service.
type KMSEncryptAPI interface {
    Encrypt(ctx context.Context,
        params *kms.EncryptInput,
        optFns ...func(*kms.Options)) (*kms.EncryptOutput, error)
}
```

```
// EncryptText encrypts some text using an AWS Key Management Service (AWS KMS) key
// (KMS key).
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If success, an EncryptOutput object containing the result of the service call
//   and nil.
//   Otherwise, nil and an error from the call to Encrypt.
func EncryptText(c context.Context, api KMSEncryptAPI, input *kms.EncryptInput)
(*kms.EncryptOutput, error) {
    return api.Encrypt(c, input)
}

func main() {
    keyID := flag.String("k", "", "The ID of a KMS key")
    text := flag.String("t", "", "The text to encrypt")
    flag.Parse()

    if *keyID == "" || *text == "" {
        fmt.Println("You must supply the ID of a KMS key and some text")
        fmt.Println("-k KEY-ID -t \"text\"")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := kms.NewFromConfig(cfg)

    input := &kms.EncryptInput{
        KeyId:    keyID,
        Plaintext: []byte(*text),
    }

    result, err := EncryptText(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got error encrypting data:")
        fmt.Println(err)
        return
    }

    blobString := b64.StdEncoding.EncodeToString(result.CiphertextBlob)

    fmt.Println(blobString)
}
```

- For API details, see [Encrypt](#) in *AWS SDK for Go API Reference*.

## Recencrypt ciphertext from one key to another

The following code example shows how to reencrypt ciphertext from one KMS key to another.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/kms"
)

// KMSReEncryptAPI defines the interface for the ReEncrypt function.
// We use this interface to test the function using a mocked service.
type KMSReEncryptAPI interface {
    ReEncrypt(ctx context.Context,
        params *kms.ReEncryptInput,
        optFns ...func(*kms.Options)) (*kms.ReEncryptOutput, error)
}

// ReEncryptText reencrypts some text using a new AWS Key Management Service (AWS KMS)
// key (KMS key).
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If success, a ReEncryptOutput object containing the result of the service call
//     and nil.
//     Otherwise, nil and an error from the call to ReEncrypt.
func ReEncryptText(c context.Context, api KMSReEncryptAPI, input *kms.ReEncryptInput)
    (*kms.ReEncryptOutput, error) {
    return api.ReEncrypt(c, input)
}

func main() {
    keyID := flag.String("k", "", "The ID of a KMS key")
    data := flag.String("d", "", "The data to reencrypt, as a string")
    flag.Parse()

    if *keyID == "" || *data == "" {
        fmt.Println("You must supply the ID of a KMS key and data")
        fmt.Println("-k KEY-ID -d DATA")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := kms.NewFromConfig(cfg)

    blob := []byte(*data)

    input := &kms.ReEncryptInput{
        CiphertextBlob: blob,
        DestinationKeyId: keyID,
    }

    result, err := ReEncryptText(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got error reencrypting data:")
        fmt.Println(err)
        return
    }
}
```

```
    }

    fmt.Println("Blob (base-64 byte array):")
    fmt.Println(result.CiphertextBlob)
}
```

- For API details, see [ReEncrypt](#) in *AWS SDK for Go API Reference*.

## Amazon S3 examples using SDK for Go V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Go V2 with Amazon Simple Storage Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2778\)](#)
- [Scenarios \(p. 2784\)](#)

## Actions

### Copy an object from one bucket to another

The following code example shows how to copy an S3 object from one bucket to another.

#### SDK for Go V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Copy an object to another name.

// CopyObject is "Pull an object from the source bucket + path".
// The semantics of CopySource varies depending on whether you're using Amazon S3 on
// Outposts,
// or through access points.
// See https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CopyObject.html#API_CopyObject_RequestSyntax
fmt.Println("Copy an object from another bucket to our bucket.")
_, err = client.CopyObject(context.TODO(), &s3.CopyObjectInput{
    Bucket:    aws.String(name),
    CopySource: aws.String(name + "/path/myfile.jpg"),
    Key:       aws.String("other/file.jpg"),
})

if err != nil {
    panic("Couldn't copy the object to a new key")
}
```

- For API details, see [CopyObject](#) in *AWS SDK for Go API Reference*.

## Create a bucket

The following code example shows how to create an S3 bucket.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Create a bucket: We're going to create a bucket to hold content.  
// Best practice is to use the preset private access control list (ACL).  
// If you are not creating a bucket from us-east-1, you must specify a bucket location  
constraint.  
// Bucket names must conform to several rules; read more at https://  
docs.aws.amazon.com/AmazonS3/latest/userguide/bucketnamingrules.html  
_, err := client.CreateBucket(context.TODO(), &s3.CreateBucketInput{  
    Bucket: aws.String(name),  
    ACL: types.BucketCannedACLPrivate,  
    CreateBucketConfiguration: &types.CreateBucketConfiguration{LocationConstraint:  
        types.BucketLocationConstraintUsWest2},  
})  
  
if err != nil {  
    panic("could not create bucket: " + err.Error())  
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for Go API Reference*.

## Delete an empty bucket

The following code example shows how to delete an empty S3 bucket.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
fmt.Println("Delete a bucket")  
// Delete the bucket.  
  
_, err = client.DeleteBucket(context.TODO(), &s3.DeleteBucketInput{  
    Bucket: aws.String(name),  
})  
if err != nil {  
    panic("Couldn't delete bucket: " + err.Error())  
}
```

- For API details, see [DeleteBucket](#) in *AWS SDK for Go API Reference*.

## Delete an object

The following code example shows how to delete an S3 object.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Delete a single object.  
fmt.Println("Delete an object from a bucket")  
, err := client.DeleteObject(context.TODO(), &s3.DeleteObjectInput{  
    Bucket: aws.String(name),  
    Key:    aws.String("other/file.jpg"),  
})  
if err != nil {  
    panic("Couldn't delete object!")  
}
```

- For API details, see [DeleteObject](#) in *AWS SDK for Go API Reference*.

## Get the ACL of a bucket

The following code example shows how to get the access control list (ACL) of an S3 bucket.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main  
  
import (  
    "context"  
    "flag"  
    "fmt"  
  
    "github.com/aws/aws-sdk-go-v2/config"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
)  
  
// S3GetBucketAclAPI defines the interface for the GetBucketAcl function.  
// We use this interface to test the function using a mocked service.  
type S3GetBucketAclAPI interface {  
    GetBucketAcl(ctx context.Context,  
        params *s3.GetBucketAclInput,  
        optFns ...func(*s3.Options)) (*s3.GetBucketAclOutput, error)  
}  
  
// FindBucketAcl retrieves the access control list (ACL) for an Amazon Simple Storage  
// Service (Amazon S3) bucket.  
// Inputs:  
//     c is the context of the method call, which includes the AWS Region  
//     api is the interface that defines the method call  
//     input defines the input arguments to the service call.  
// Output:  
//     If success, a GetBucketAclOutput object containing the result of the service  
//     call and nil  
//     Otherwise, nil and an error from the call to GetBucketAcl  
func FindBucketAcl(c context.Context, api S3GetBucketAclAPI, input  
    *s3.GetBucketAclInput) (*s3.GetBucketAclOutput, error) {
```

```
    return api.GetBucketAcl(c, input)
}

func main() {
    bucket := flag.String("b", "", "The bucket for which the ACL is returned")
    flag.Parse()

    if *bucket == "" {
        fmt.Println("You must supply a bucket name (-b BUCKET)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := s3.NewFromConfig(cfg)

    input := &s3.GetBucketAclInput{
        Bucket: bucket,
    }

    result, err := FindBucketAcl(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error retrieving ACL for " + *bucket)
        return
    }

    fmt.Println("Owner:", *result.Owner.DisplayName)
    fmt.Println("")
    fmt.Println("Grants")

    for _, g := range result.Grants {
        // If we add a canned ACL, the name is nil
        if g.Grantee.DisplayName == nil {
            fmt.Println(" Grantee: EVERYONE")
        } else {
            fmt.Println(" Grantee: ", *g.Grantee.DisplayName)
        }

        fmt.Println(" Type: ", string(g.Grantee.Type))
        fmt.Println(" Permission: ", string(g.Permission))
        fmt.Println("")
    }
}
```

- For API details, see [GetBucketAcl](#) in *AWS SDK for Go API Reference*.

## Get the ACL of an object

The following code example shows how to get the access control list (ACL) of an S3 object.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main
```

```
import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

// S3GetObjectAclAPI defines the interface for the GetObjectAcl function.
// We use this interface to test the function using a mocked service.
type S3GetObjectAclAPI interface {
    GetObjectAcl(ctx context.Context,
        params *s3.GetObjectAclInput,
        optFns ...func(*s3.Options)) (*s3.GetObjectAclOutput, error)
}

// FindObjectAcl gets the access control list (ACL) for an Amazon Simple Storage
// Service (Amazon S3) bucket object
// Inputs:
//     c is the context of the method call, which includes the AWS Region
//     api is the interface that defines the method call
//     input defines the input arguments to the service call.
// Output:
//     If success, a GetObjectAclOutput object containing the result of the service
//     call and nil
//     Otherwise, nil and an error from the call to GetObjectAcl
func FindObjectAcl(c context.Context, api S3GetObjectAclAPI, input
    *s3.GetObjectAclInput) (*s3.GetObjectAclOutput, error) {
    return api.GetObjectAcl(c, input)
}

func main() {
    bucket := flag.String("b", "", "The bucket containing the object")
    objectName := flag.String("o", "", "The bucket object to get ACL from")
    flag.Parse()

    if *bucket == "" || *objectName == "" {
        fmt.Println("You must supply a bucket (-b BUCKET) and object (-o OBJECT)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := s3.NewFromConfig(cfg)

    input := &s3.GetObjectAclInput{
        Bucket: bucket,
        Key:    objectName,
    }

    result, err := FindObjectAcl(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error getting ACL for " + *objectName)
        return
    }

    fmt.Println("Owner:", *result.Owner.DisplayName)
    fmt.Println("")
    fmt.Println("Grants")

    for _, g := range result.Grants {
        fmt.Println("  Grantee:  ", *g.Grantee.DisplayName)
    }
}
```

```
    fmt.Println("  Type:      ", string(g.Grantee.Type))
    fmt.Println("  Permission:", string(g.Permission))
    fmt.Println("")
}
```

- For API details, see [GetObjectAcl](#) in *AWS SDK for Go API Reference*.

## List buckets

The following code example shows how to list S3 buckets.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
listBucketsResult, err := client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})

if err != nil {
    panic("Couldn't list buckets")
}

for _, bucket := range listBucketsResult.Buckets {
    fmt.Printf("Bucket name: %s\t\tcreated at: %v\n", *bucket.Name, bucket.CreationDate)
}
```

- For API details, see [ListBuckets](#) in *AWS SDK for Go API Reference*.

## List objects in a bucket

The following code example shows how to list objects in an S3 bucket.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// List objects in the bucket.
// n.b. object keys in Amazon S3 do not begin with '/'. You do not need to lead your
// prefix with it.
fmt.Println("Listing the objects in the bucket:")
listObjsResponse, err := client.ListObjectsV2(context.TODO(), &s3.ListObjectsV2Input{
    Bucket: aws.String(name),
    Prefix: aws.String(""),
})

if err != nil {
    panic("Couldn't list bucket contents")
}

for _, object := range listObjsResponse.Contents {
    fmt.Printf("%s (%d bytes, class %v) \n", *object.Key, object.Size,
    object.StorageClass)
}
```

- For API details, see [ListObjects](#) in *AWS SDK for Go API Reference*.

## Upload an object to a bucket

The following code example shows how to upload an object to an S3 bucket.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Place an object in a bucket.
fmt.Println("Upload an object to the bucket")
// Get the object body to upload.
// Image credit: https://unsplash.com/photos/iz58d89q3ss
stat, err := os.Stat("image.jpg")
if err != nil {
    panic("Couldn't stat image: " + err.Error())
}
file, err := os.Open("image.jpg")

if err != nil {
    panic("Couldn't open local file")
}

_, err = client.PutObject(context.TODO(), &s3.PutObjectInput{
    Bucket:      aws.String(name),
    Key:         aws.String("path/myfile.jpg"),
    Body:         file,
    ContentLength: stat.Size(),
})
file.Close()

if err != nil {
    panic("Couldn't upload file: " + err.Error())
}
```

- For API details, see [PutObject](#) in *AWS SDK for Go API Reference*.

## Scenarios

### Create a presigned URL

The following code example shows how to create a presigned URL for S3 and upload an object.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get a presigned URL for the object.
// In order to get a presigned URL for an object, you must
```

```
// create a Presignclient
fmt.Println("Create Presign client")
presignClient := s3.NewPresignClient(&client)

presignParams := &s3.GetObjectInput{
    Bucket: aws.String(name),
    Key:    aws.String("path/myfile.jpg"),
}

// Apply an expiration via an option function
presignDuration := func(po *s3.PresignOptions) {
    po.Expires = 5 * time.Minute
}

presignResult, err := presignClient.PresignGetObject(context.TODO(), presignParams,
presignDuration)

if err != nil {
    panic("Couldn't get presigned URL for GetObject")
}

fmt.Printf("Presigned URL For object: %s\n", presignResult.URL)
```

## Get started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.
- Delete a bucket.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// This bucket name is 100% unique.
// Remember that bucket names must be globally unique among all buckets.

myBucketName := "mybucket-" + (xid.New().String())
fmt.Printf("Bucket name: %v\n", myBucketName)

cfg, err := config.LoadDefaultConfig(context.TODO())

if err != nil {
    panic("Failed to load configuration")
}

s3client := s3.NewFromConfig(cfg)

MakeBucket(*s3client, myBucketName)
```

```
BucketOps(*s3client, myBucketName)
AccountBucketOps(*s3client, myBucketName)
BucketDelOps(*s3client, myBucketName)
```

- For API details, see the following topics in *AWS SDK for Go API Reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

## Amazon SNS examples using SDK for Go V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Go V2 with Amazon Simple Notification Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2786\)](#)

## Actions

### Create a topic

The following code example shows how to create an Amazon SNS topic.

#### SDK for Go V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

- For API details, see [CreateTopic](#) in *AWS SDK for Go API Reference*.

### List the subscribers of a topic

The following code example shows how to retrieve the list of subscribers of an Amazon SNS topic.

#### SDK for Go V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

- For API details, see [ListSubscriptions](#) in *AWS SDK for Go API Reference*.

### List topics

The following code example shows how to list Amazon SNS topics.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

- For API details, see [ListTopics](#) in *AWS SDK for Go API Reference*.

## Publish to a topic

The following code example shows how to publish messages to an Amazon SNS topic.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

- For API details, see [Publish](#) in *AWS SDK for Go API Reference*.

## Subscribe an email address to a topic

The following code example shows how to subscribe an email address to an Amazon SNS topic.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

- For API details, see [Subscribe](#) in *AWS SDK for Go API Reference*.

## Amazon SQS examples using SDK for Go V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Go V2 with Amazon Simple Queue Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2787\)](#)

## Actions

### Change message timeout visibility

The following code example shows how to change an Amazon SQS message timeout visibility.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main
```

```
import (
    "context"
    "flag"
    "fmt"
    "strconv"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// SQSSetMsgVisibilityAPI defines the interface for the GetQueueUrl and
// ChangeMessageVisibility functions.
// We use this interface to test the functions using a mocked service.
type SQSSetMsgVisibilityAPI interface {
    GetQueueUrl(ctx context.Context,
        params *sqs.GetQueueUrlInput,
        optFns ...func(*sqs.Options)) (*sqs.GetQueueUrlOutput, error)

    ChangeMessageVisibility(ctx context.Context,
        params *sqs.ChangeMessageVisibilityInput,
        optFns ...func(*sqs.Options)) (*sqs.ChangeMessageVisibilityOutput, error)
}

// GetQueueURL gets the URL of an Amazon SQS queue.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If success, a GetQueueUrlOutput object containing the result of the service call
//     and nil.
//     Otherwise, nil and an error from the call to GetQueueUrl.
func GetQueueURL(c context.Context, api SQSSetMsgVisibilityAPI, input
    *sqs.GetQueueUrlInput) (*sqs.GetQueueUrlOutput, error) {
    return api.GetQueueUrl(c, input)
}

// SetMsgVisibility sets the visibility timeout for a message in an Amazon SQS queue.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If success, a ChangeMessageVisibilityOutput object containing the result of the
//     service call and nil.
//     Otherwise, nil and an error from the call to ChangeMessageVisibility.
func SetMsgVisibility(c context.Context, api SQSSetMsgVisibilityAPI, input
    *sqs.ChangeMessageVisibilityInput) (*sqs.ChangeMessageVisibilityOutput, error) {
    return api.ChangeMessageVisibility(c, input)
}

func main() {
    queue := flag.String("q", "", "The name of the queue")
    handle := flag.String("h", "", "The receipt handle of the message")
    visibility := flag.Int("v", 30, "The duration, in seconds, that the message is not
        visible to other consumers")
    flag.Parse()

    if *queue == "" || *handle == "" {
        fmt.Println("You must supply a queue name (-q QUEUE) and message receipt handle (-h
        HANDLE)")
        return
    }
}
```

```
if *visibility < 0 {
    *visibility = 0
}

if *visibility > 12*60*60 {
    *visibility = 12 * 60 * 60
}

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic("configuration error, " + err.Error())
}

client := sqs.NewFromConfig(cfg)

gQInput := &sqs.GetQueueUrlInput{
    QueueName: queue,
}

// Get URL of queue
urlResult, err := GetQueueURL(context.TODO(), client, gQInput)
if err != nil {
    fmt.Println("Got an error getting the queue URL:")
    fmt.Println(err)
    return
}

queueURL := urlResult.QueueUrl

sVInput := &sqs.ChangeMessageVisibilityInput{
    ReceiptHandle:     handle,
    QueueUrl:         queueURL,
    VisibilityTimeout: int32(*visibility),
}

_, err = SetMsgVisibility(context.TODO(), client, sVInput)
if err != nil {
    fmt.Println("Got an error setting the visibility of the message:")
    fmt.Println(err)
    return
}

fmt.Println("Changed the visibility of the message with the handle " + *handle + " in
the " + *queue + " to " + strconv.Itoa(*visibility))
}
```

- For API details, see [ChangeMessageVisibility](#) in *AWS SDK for Go API Reference*.

## Create a queue

The following code example shows how to create an Amazon SQS queue.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main
```

```
import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// SQSCreateQueueAPI defines the interface for the CreateQueue function.
// We use this interface to test the function using a mocked service.
type SQSCreateQueueAPI interface {
    CreateQueue(ctx context.Context,
        params *sqs.CreateQueueInput,
        optFns ...func(*sqs.Options)) (*sqs.CreateQueueOutput, error)
}

// CreateQueue creates an Amazon SQS queue.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If success, a CreateQueueOutput object containing the result of the service call
//     and nil.
//     Otherwise, nil and an error from the call to CreateQueue.
func CreateQueue(c context.Context, api SQSCreateQueueAPI, input *sqs.CreateQueueInput)
    (*sqs.CreateQueueOutput, error) {
    return api.CreateQueue(c, input)
}

func main() {
    queue := flag.String("q", "", "The name of the queue")
    flag.Parse()

    if *queue == "" {
        fmt.Println("You must supply a queue name (-q QUEUE)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := sqs.NewFromConfig(cfg)

    input := &sqs.CreateQueueInput{
        QueueName: queue,
        Attributes: map[string]string{
            "DelaySeconds":           "60",
            "MessageRetentionPeriod": "86400",
        },
    }

    result, err := CreateQueue(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error creating the queue:")
        fmt.Println(err)
        return
    }

    fmt.Println("URL: " + *result.QueueUrl)
}
```

- For API details, see [CreateQueue](#) in *AWS SDK for Go API Reference*.

## Delete a message from a queue

The following code example shows how to delete a message from an Amazon SQS queue.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// SQSDelateMessageAPI defines the interface for the GetQueueUrl and DeleteMessage
// functions.
// We use this interface to test the functions using a mocked service.
type SQSDelateMessageAPI interface {
    GetQueueUrl(ctx context.Context,
        params *sqs.GetQueueUrlInput,
        optFns ...func(*sqs.Options)) (*sqs.GetQueueUrlOutput, error)

    DeleteMessage(ctx context.Context,
        params *sqs.DeleteMessageInput,
        optFns ...func(*sqs.Options)) (*sqs.DeleteMessageOutput, error)
}

// GetQueueURL gets the URL of an Amazon SQS queue.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If success, a GetQueueUrlOutput object containing the result of the service call
//     and nil.
//     Otherwise, nil and an error from the call to GetQueueUrl.
func GetQueueURL(c context.Context, api SQSDelateMessageAPI, input
    *sqs.GetQueueUrlInput) (*sqs.GetQueueUrlOutput, error) {
    return api.GetQueueUrl(c, input)
}

// RemoveMessage deletes a message from an Amazon SQS queue.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If success, a DeleteMessageOutput object containing the result of the service
//     call and nil.
//     Otherwise, nil and an error from the call to DeleteMessage.
func RemoveMessage(c context.Context, api SQSDelateMessageAPI, input
    *sqs.DeleteMessageInput) (*sqs.DeleteMessageOutput, error) {
```

```
    return api.DeleteMessage(c, input)
}

func main() {
    queue := flag.String("q", "", "The name of the queue")
    messageHandle := flag.String("m", "", "The receipt handle of the message")
    flag.Parse()

    if *queue == "" || *messageHandle == "" {
        fmt.Println("You must supply a queue name (-q QUEUE) and message receipt handle (-m MESSAGE-HANDLE)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := sqs.NewFromConfig(cfg)

    qUIInput := &sqs.GetQueueUrlInput{
        QueueName: queue,
    }

    // Get URL of queue
    result, err := GetQueueURL(context.TODO(), client, qUIInput)
    if err != nil {
        fmt.Println("Got an error getting the queue URL:")
        fmt.Println(err)
        return
    }

    queueURL := result.QueueUrl

    dMInput := &sqs.DeleteMessageInput{
        QueueUrl: queueURL,
        ReceiptHandle: messageHandle,
    }

    _, err = RemoveMessage(context.TODO(), client, dMInput)
    if err != nil {
        fmt.Println("Got an error deleting the message:")
        fmt.Println(err)
        return
    }

    fmt.Println("Deleted message from queue with URL " + *queueURL)
}
```

- For API details, see [DeleteMessage](#) in [AWS SDK for Go API Reference](#).

## Delete a queue

The following code example shows how to delete an Amazon SQS queue.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an Amazon SQS queue.

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// SQSDelQueueAPI defines the interface for the GetQueueUrl and DeleteQueue
// functions.
// We use this interface to test the functions using a mocked service.
type SQSDelQueueAPI interface {
    GetQueueUrl(ctx context.Context,
        params *sqs.GetQueueUrlInput,
        optFns ...func(*sqs.Options)) (*sqs.GetQueueUrlOutput, error)

    DeleteQueue(ctx context.Context,
        params *sqs.DeleteQueueInput,
        optFns ...func(*sqs.Options)) (*sqs.DeleteQueueOutput, error)
}

// GetQueueURL gets the URL of an Amazon SQS queue.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If success, a GetQueueUrlOutput object containing the result of the service call
//   and nil.
//   Otherwise, nil and an error from the call to GetQueueUrl.
func GetQueueURL(c context.Context, api SQSDelQueueAPI, input *sqs.GetQueueUrlInput)
    (*sqs.GetQueueUrlOutput, error) {
    return api.GetQueueUrl(c, input)
}

// DeleteQueue deletes an Amazon SQS queue.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If success, a DeleteQueueOutput object containing the result of the service call
//   and nil.
//   Otherwise, nil and an error from the call to DeleteQueue.
func DeleteQueue(c context.Context, api SQSDelQueueAPI, input *sqs.DeleteQueueInput)
    (*sqs.DeleteQueueOutput, error) {
    return api.DeleteQueue(c, input)
}

func main() {
    queue := flag.String("q", "", "The name of the queue")
    flag.Parse()

    if *queue == "" {
        fmt.Println("You must supply a queue name (-q QUEUE")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
```

```
    panic("configuration error, " + err.Error())
}

client := sqs.NewFromConfig(cfg)

qInput := &sqs.GetQueueUrlInput{
    QueueName: queue,
}

// Get the URL for the queue
result, err := GetQueueURL(context.TODO(), client, qInput)
if err != nil {
    fmt.Println("Got an error getting the queue URL:")
    fmt.Println(err)
    return
}

queueURL := result.QueueUrl

dqInput := &sqs.DeleteQueueInput{
    QueueUrl: queueURL,
}

_, err = DeleteQueue(context.TODO(), client, dqInput)
if err != nil {
    fmt.Println("Got an error deleting the queue:")
    fmt.Println(err)
    return
}

fmt.Println("Deleted queue with URL " + *queueURL)
}
```

- For API details, see [DeleteQueue](#) in *AWS SDK for Go API Reference*.

## Get the URL of a queue

The following code example shows how to get the URL of an Amazon SQS queue.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get the URL for an Amazon SQS queue.

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// SQSGetQueueUrlAPI defines the interface for the GetQueueUrl function.
// We use this interface to test the function using a mocked service.
type SQSGetQueueUrlAPI interface {
    GetQueueUrl(ctx context.Context,
```

```
    params *sqsc.GetQueueUrlInput,
    optFns ...func(*sqsc.Options)) (*sqsc.GetQueueUrlOutput, error)
}

// GetQueueURL gets the URL of an Amazon SQS queue.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If success, a GetQueueUrlOutput object containing the result of the service call
//   and nil.
//   Otherwise, nil and an error from the call to GetQueueUrl.
func GetQueueURL(c context.Context, api SQSGetQueueUrlAPI, input *sqsc.GetQueueUrlInput)
(*sqsc.GetQueueUrlOutput, error) {
    return api.GetQueueUrl(c, input)
}

func main() {
    queue := flag.String("q", "", "The name of the queue")
    flag.Parse()

    if *queue == "" {
        fmt.Println("You must supply a queue name (-q QUEUE)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := sqs.NewFromConfig(cfg)

    input := &sqsc.GetQueueUrlInput{
        QueueName: queue,
    }

    result, err := GetQueueURL(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error getting the queue URL:")
        fmt.Println(err)
        return
    }

    fmt.Println("URL: " + *result.QueueUrl)
}
```

- For API details, see [GetQueueUrl](#) in *AWS SDK for Go API Reference*.

## List queues

The following code example shows how to list Amazon SQS queues.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List your Amazon SQS queues.

```
package main

import (
    "context"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// SQSListQueuesAPI defines the interface for the ListQueues function.
// We use this interface to test the function using a mocked service.
type SQSListQueuesAPI interface {
    ListQueues(ctx context.Context,
        params *sqs.ListQueuesInput,
        optFns ...func(*sqs.Options)) (*sqs.ListQueuesOutput, error)
}

// GetQueues retrieves a list of your Amazon Simple Queue Service (Amazon SQS) queues.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If success, a ListQueuesOutput object containing the result of the service call
//     and nil.
//     Otherwise, nil and an error from the call to ListQueues.
func GetQueues(c context.Context, api SQSListQueuesAPI, input *sqs.ListQueuesInput) (*sqs.ListQueuesOutput, error) {
    return api.ListQueues(c, input)
}

func main() {
    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := sqs.NewFromConfig(cfg)

    input := &sqs.ListQueuesInput{}

    result, err := GetQueues(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error retrieving queue URLs:")
        fmt.Println(err)
        return
    }

    for i, url := range result.QueueUrls {
        fmt.Printf("%d: %s\n", i+1, url)
    }
}
```

- For API details, see [ListQueues](#) in *AWS SDK for Go API Reference*.

## Receive messages from a queue

The following code example shows how to receive messages from an Amazon SQS queue.

## SDK for Go V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Receive a message from an Amazon SQS queue.

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SQSReceiveMessageAPI defines the interface for the GetQueueUrl function.
// We use this interface to test the function using a mocked service.
type SQSReceiveMessageAPI interface {
    GetQueueUrl(ctx context.Context,
        params *sqs.GetQueueUrlInput,
        optFns ...func(*sqs.Options)) (*sqs.GetQueueUrlOutput, error)

    ReceiveMessage(ctx context.Context,
        params *sqs.ReceiveMessageInput,
        optFns ...func(*sqs.Options)) (*sqs.ReceiveMessageOutput, error)
}

// GetQueueURL gets the URL of an Amazon SQS queue.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If success, a GetQueueUrlOutput object containing the result of the service call
//   and nil.
//   Otherwise, nil and an error from the call to GetQueueUrl.
func GetQueueURL(c context.Context, api SQSReceiveMessageAPI, input
    *sqs.GetQueueUrlInput) (*sqs.GetQueueUrlOutput, error) {
    return api.GetQueueUrl(c, input)
}

// GetMessages gets the most recent message from an Amazon SQS queue.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If success, a ReceiveMessageOutput object containing the result of the service
//   call and nil.
//   Otherwise, nil and an error from the call to ReceiveMessage.
func GetMessages(c context.Context, api SQSReceiveMessageAPI, input
    *sqs.ReceiveMessageInput) (*sqs.ReceiveMessageOutput, error) {
    return api.ReceiveMessage(c, input)
}

func main() {
    queue := flag.String("q", "", "The name of the queue")
    timeout := flag.Int("t", 5, "How long, in seconds, that the message is hidden from
        others")
    flag.Parse()
}
```

```
if *queue == "" {
    fmt.Println("You must supply the name of a queue (-q QUEUE)")
    return
}

if *timeout < 0 {
    *timeout = 0
}

if *timeout > 12*60*60 {
    *timeout = 12 * 60 * 60
}

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic("configuration error, " + err.Error())
}

client := sqs.NewFromConfig(cfg)

gQInput := &sqs.GetQueueUrlInput{
    QueueName: queue,
}

// Get URL of queue
urlResult, err := GetQueueURL(context.TODO(), client, gQInput)
if err != nil {
    fmt.Println("Got an error getting the queue URL:")
    fmt.Println(err)
    return
}

queueURL := urlResult.QueueUrl

gMInput := &sqs.ReceiveMessageInput{
    MessageAttributeNames: []string{
        string(types.QueueAttributeNameAll),
    },
    QueueUrl:           queueURL,
    MaxNumberOfMessages: 1,
    VisibilityTimeout:   int32(*timeout),
}

msgResult, err := GetMessages(context.TODO(), client, gMInput)
if err != nil {
    fmt.Println("Got an error receiving messages:")
    fmt.Println(err)
    return
}

if msgResult.Messages != nil {
    fmt.Println("Message ID: " + *msgResult.Messages[0].MessageId)
    fmt.Println("Message Handle: " + *msgResult.Messages[0].ReceiptHandle)
} else {
    fmt.Println("No messages found")
}
```

- For API details, see [ReceiveMessage](#) in *AWS SDK for Go API Reference*.

## Send a message to a queue

The following code example shows how to send a message to an Amazon SQS queue.

### SDK for Go V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Send a message to an Amazon SQS queue.

```
package main

import (
    "context"
    "flag"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SQSSendMessageAPI defines the interface for the GetQueueUrl and SendMessage
// functions.
// We use this interface to test the functions using a mocked service.
type SQSSendMessageAPI interface {
    GetQueueUrl(ctx context.Context,
        params *sqs.GetQueueUrlInput,
        optFns ...func(*sqs.Options)) (*sqs.GetQueueUrlOutput, error)

    SendMessage(ctx context.Context,
        params *sqs.SendMessageInput,
        optFns ...func(*sqs.Options)) (*sqs.SendMessageOutput, error)
}

// GetQueueURL gets the URL of an Amazon SQS queue.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If success, a GetQueueUrlOutput object containing the result of the service call
//     and nil.
//     Otherwise, nil and an error from the call to GetQueueUrl.
func GetQueueURL(c context.Context, api SQSSendMessageAPI, input *sqs.GetQueueUrlInput)
    (*sqs.GetQueueUrlOutput, error) {
    return api.GetQueueUrl(c, input)
}

// SendMsg sends a message to an Amazon SQS queue.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If success, a SendMessageOutput object containing the result of the service call
//     and nil.
//     Otherwise, nil and an error from the call to SendMessage.
func SendMsg(c context.Context, api SQSSendMessageAPI, input *sqs.SendMessageInput)
    (*sqs.SendMessageOutput, error) {
    return api.SendMessage(c, input)
}
```

```
func main() {
    queue := flag.String("q", "", "The name of the queue")
    flag.Parse()

    if *queue == "" {
        fmt.Println("You must supply the name of a queue (-q QUEUE)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := sqs.NewFromConfig(cfg)

    // Get URL of queue
    gQInput := &sqs.GetQueueUrlInput{
        QueueName: queue,
    }

    result, err := GetQueueURL(context.TODO(), client, gQInput)
    if err != nil {
        fmt.Println("Got an error getting the queue URL:")
        fmt.Println(err)
        return
    }

    queueURL := result.QueueUrl

    sMInput := &sqs.SendMessageInput{
        DelaySeconds: 10,
        MessageAttributes: map[string]types.MessageAttributeValue{
            "Title": {
                DataType: aws.String("String"),
                StringValue: aws.String("The Whistler"),
            },
            "Author": {
                DataType: aws.String("String"),
                StringValue: aws.String("John Grisham"),
            },
            "WeeksOn": {
                DataType: aws.String("Number"),
                StringValue: aws.String("6"),
            },
        },
        MessageBody: aws.String("Information about the NY Times fiction bestseller for the week of 12/11/2016."),
        QueueUrl: queueURL,
    }

    resp, err := SendMsg(context.TODO(), client, sMInput)
    if err != nil {
        fmt.Println("Got an error sending the message:")
        fmt.Println(err)
        return
    }

    fmt.Println("Sent message with ID: " + *resp.MessageId)
}
```

- For API details, see [SendMessage](#) in *AWS SDK for Go API Reference*.

# Code examples for SDK for JavaScript V2

The code examples in this topic show you how to use the AWS SDK for JavaScript V2 with AWS.

## Examples

- [Single-service actions and scenarios using SDK for JavaScript V2 \(p. 2801\)](#)
- [Cross-service examples using SDK for JavaScript V2 \(p. 2909\)](#)

## Single-service actions and scenarios using SDK for JavaScript V2

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V2 with AWS services.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Services

- [CloudWatch examples using SDK for JavaScript V2 \(p. 2801\)](#)
- [CloudWatch Events examples using SDK for JavaScript V2 \(p. 2813\)](#)
- [CloudWatch Logs examples using SDK for JavaScript V2 \(p. 2817\)](#)
- [DynamoDB examples using SDK for JavaScript V2 \(p. 2822\)](#)
- [EventBridge examples using SDK for JavaScript V2 \(p. 2847\)](#)
- [IAM examples using SDK for JavaScript V2 \(p. 2851\)](#)
- [Amazon Pinpoint examples using SDK for JavaScript V2 \(p. 2878\)](#)
- [Amazon Pinpoint SMS and Voice API examples using SDK for JavaScript V2 \(p. 2885\)](#)
- [S3 Glacier examples using SDK for JavaScript V2 \(p. 2887\)](#)
- [Amazon SNS examples using SDK for JavaScript V2 \(p. 2890\)](#)
- [Amazon SQS examples using SDK for JavaScript V2 \(p. 2892\)](#)
- [AWS STS examples using SDK for JavaScript V2 \(p. 2907\)](#)

## CloudWatch examples using SDK for JavaScript V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V2 with Amazon CloudWatch.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Topics

- [Actions \(p. 2802\)](#)

## Actions

### Create an alarm that watches a metric

The following code example shows how to create an Amazon CloudWatch alarm that watches a metric.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { PutMetricAlarmCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanOrEqualToThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: false,
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

export const run = async () => {
  try {
    const data = await cwClient.send(new PutMetricAlarmCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutMetricAlarm in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
    AlarmName: 'Web_Server_CPU_Utilization',
    ComparisonOperator: 'GreaterThanOrEqualToThreshold',
    EvaluationPeriods: 1,
    MetricName: 'CPUUtilization',
    Namespace: 'AWS/EC2',
    Period: 60,
    Statistic: 'Average',
    Threshold: 70.0,
    ActionsEnabled: false,
    AlarmDescription: 'Alarm when server CPU exceeds 70%',
    Dimensions: [
        {
            Name: 'InstanceId',
            Value: 'INSTANCE_ID'
        },
    ],
    Unit: 'Percent'
};

cw.putMetricAlarm(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutMetricAlarm in AWS SDK for JavaScript API Reference](#).

## Delete alarms

The following code example shows how to delete Amazon CloudWatch alarms.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { DeleteAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = { AlarmNames: "ALARM_NAME" }; // e.g.,
  "Web_Server_CPU_Utilization"

export const run = async () => {
  try {
    const data = await cwClient.send(new DeleteAlarmsCommand(params));
    console.log("Success, alarm deleted; requestID:", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};

// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAlarms](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Import the SDK and client modules and call the API.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
  AlarmNames: ['Web_Server_CPU_Utilization']
};

cw.deleteAlarms(params, function(err, data) {
  if (err) {
```

```
    console.log("Error", err);
} else {
    console.log("Success", data);
}
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAlarms](#) in *AWS SDK for JavaScript API Reference*.

## Describe alarms for a metric

The following code example shows how to describe Amazon CloudWatch alarms for a metric.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { DescribeAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = { StateValue: "INSUFFICIENT_DATA" };

export const run = async () => {
    try {
        const data = await cwClient.send(new DescribeAlarmsCommand(params));
        console.log("Success", data);
        data.MetricAlarms.forEach(function (item) {
            console.log(item.AlarmName);
        });
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeAlarmsForMetric](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

cw.describeAlarms({StateValue: 'INSUFFICIENT_DATA'}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    // List the names of all current alarms in the console
    data.MetricAlarms.forEach(function (item, index, array) {
      console.log(item.AlarmName);
    });
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeAlarmsForMetric](#) in [AWS SDK for JavaScript API Reference](#).

## Disable alarm actions

The following code example shows how to disable Amazon CloudWatch alarm actions.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { DisableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = { AlarmNames: "ALARM_NAME" }; // e.g.,
  "Web_Server_CPU_Utilization"
```

```
export const run = async () => {
  try {
    const data = await cwClient.send(new DisableAlarmActionsCommand(params));
    console.log("Success, alarm disabled:", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DisableAlarmActions](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Import the SDK and client modules and call the API.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

cw.disableAlarmActions({AlarmNames: ['Web_Server_CPU_Utilization']}, function(err,
data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DisableAlarmActions](#) in [AWS SDK for JavaScript API Reference](#).

### Enable alarm actions

The following code example shows how to enable Amazon CloudWatch alarm actions.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
```

```
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {
  PutMetricAlarmCommand,
  EnableAlarmActionsCommand,
} from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = {
  AlarmName: "ALARM_NAME", //ALARM_NAME
  ComparisonOperator: "GreaterThanOrEqualToThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: true,
  AlarmActions: ["ACTION_ARN"], //e.g., "arn:aws:automate:us-east-1:ec2:stop"
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

export const run = async () => {
  try {
    const data = await cwClient.send(new PutMetricAlarmCommand(params));
    console.log("Alarm action added; RequestID:", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
  try {
    const paramsEnableAlarmAction = {
      AlarmNames: [params.AlarmName],
    };
    const data = await cwClient.send(
      new EnableAlarmActionsCommand(paramsEnableAlarmAction)
    );
    console.log("Alarm action enabled; RequestID:", data.$metadata.requestId);
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [EnableAlarmActions](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Import the SDK and client modules and call the API.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
    AlarmName: 'Web_Server_CPU_Utilization',
    ComparisonOperator: 'GreaterThanOrEqualToThreshold',
    EvaluationPeriods: 1,
    MetricName: 'CPUUtilization',
    Namespace: 'AWS/EC2',
    Period: 60,
    Statistic: 'Average',
    Threshold: 70.0,
    ActionsEnabled: true,
    AlarmActions: ['ACTION_ARN'],
    AlarmDescription: 'Alarm when server CPU exceeds 70%',
    Dimensions: [
        {
            Name: 'InstanceId',
            Value: 'INSTANCE_ID'
        },
    ],
    Unit: 'Percent'
};

cw.putMetricAlarm(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Alarm action added", data);
        var paramsEnableAlarmAction = {
            AlarmNames: [params.AlarmName]
        };
        cw.enableAlarmActions(paramsEnableAlarmAction, function(err, data) {
            if (err) {
                console.log("Error", err);
            } else {
                console.log("Alarm action enabled", data);
            }
        });
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [EnableAlarmActions](#) in *AWS SDK for JavaScript API Reference*.

## List metrics

The following code example shows how to list Amazon CloudWatch metrics.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { ListMetricsCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = {
  Dimensions: [
    {
      Name: "LogGroupName" /* required */,
    },
  ],
  MetricName: "IncomingLogEvents",
  Namespace: "AWS/Logs",
};

export const run = async () => {
  try {
    const data = await cwClient.send(new ListMetricsCommand(params));
    console.log("Success. Metrics:", JSON.stringify(data.Metrics));
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListMetrics](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});
```

```
// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
  Dimensions: [
    {
      Name: 'LogGroupName', /* required */
    },
  ],
  MetricName: 'IncomingLogEvents',
  Namespace: 'AWS/Logs'
};

cw.listMetrics(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Metrics", JSON.stringify(data.Metrics));
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListMetrics](#) in *AWS SDK for JavaScript API Reference*.

## Put data into a metric

The following code example shows how to put data into a Amazon CloudWatch metric.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { PutMetricDataCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = {
  MetricData: [
    {
      MetricName: "PAGES_VISITED",
      Dimensions: [
        {
          Name: "UNIQUE_PAGES",
          Value: "URLS",
        },
      ],
    },
  ],
};
```

```
        ],
        ],
        Unit: "None",
        Value: 1.0,
    },
],
Namespace: "SITE/TRAFFIC",
};

export const run = async () => {
    try {
        const data = await cwClient.send(new PutMetricDataCommand(params));
        console.log("Success", data.$metadata.requestId);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutMetricData](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

// Create parameters JSON for putMetricData
var params = {
    MetricData: [
        {
            MetricName: 'PAGES_VISITED',
            Dimensions: [
                {
                    Name: 'UNIQUE_PAGES',
                    Value: 'URLS'
                },
            ],
            Unit: 'None',
            Value: 1.0
        },
    ],
    Namespace: 'SITE/TRAFFIC'
};

cw.putMetricData(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", JSON.stringify(data));
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutMetricData](#) in [AWS SDK for JavaScript API Reference](#).

## CloudWatch Events examples using SDK for JavaScript V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V2 with Amazon CloudWatch Events.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2813\)](#)

## Actions

### Adding a Lambda function target

The following code example shows how to add an AWS Lambda function target to an Amazon CloudWatch Events event.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cweClient = new CloudWatchEventsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { PutTargetsCommand } from "@aws-sdk/client-cloudwatch-events";
import { cweClient } from "./libs/cloudWatchEventsClient.js";

// Set the parameters
export const params = {
  Rule: "DEMO_EVENT",
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN", //LAMBDA_FUNCTION_ARN
      Id: "myCloudWatchEventsTarget",
    },
  ],
};
```

```
export const run = async () => {
  try {
    const data = await cweClient.send(new PutTargetsCommand(params));
    console.log("Success, target added; requestID: ", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutTargets](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({apiVersion: '2015-10-07'});

var params = {
  Rule: 'DEMO_EVENT',
  Targets: [
    {
      Arn: 'LAMBDA_FUNCTION_ARN',
      Id: 'myCloudWatchEventsTarget',
    }
  ]
};

cwevents.putTargets(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutTargets](#) in [AWS SDK for JavaScript API Reference](#).

### Create a scheduled rule

The following code example shows how to create an Amazon CloudWatch Events scheduled rule.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cweClient = new CloudWatchEventsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { PutRuleCommand } from "@aws-sdk/client-cloudwatch-events";
import { cweClient } from "./libs/cloudWatchEventsClient.js";

// Set the parameters
export const params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN", //IAM_ROLE_ARN
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

export const run = async () => {
  try {
    const data = await cweClient.send(new PutRuleCommand(params));
    console.log("Success, scheduled rule created; Rule ARN:", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutRule in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({apiVersion: '2015-10-07'});

var params = {
  Name: 'DEMO_EVENT',
  RoleArn: 'IAM_ROLE_ARN',
  ScheduleExpression: 'rate(5 minutes)',
  State: 'ENABLED'
};
```

```
cwevents.putRule(params, function(err, data) {  
    if (err) {  
        console.log("Error", err);  
    } else {  
        console.log("Success", data.RuleArn);  
    }  
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutRule](#) in *AWS SDK for JavaScript API Reference*.

## Send events

The following code example shows how to send Amazon CloudWatch Events events.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create an Amazon CloudWatch service client object.  
export const cweClient = new CloudWatchEventsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js  
import { PutEventsCommand } from "@aws-sdk/client-cloudwatch-events";  
import { cweClient } from "./libs/cloudWatchEventsClient.js";  
  
// Set the parameters  
export const params = {  
    Entries: [  
        {  
            Detail: '{ "key1": "value1", "key2": "value2" }',  
            DetailType: "appRequestSubmitted",  
            Resources: [  
                "RESOURCE_ARN", //RESOURCE_ARN  
            ],  
            Source: "com.company.app",  
        },  
    ],  
};  
  
export const run = async () => {  
    try {  
        const data = await cweClient.send(new PutEventsCommand(params));  
        console.log("Success, event sent; requestID:", data);  
        return data; // For unit tests.  
    } catch (err) {  
        console.log("Error", err);  
    }  
}
```

```
};

// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutEvents](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({apiVersion: '2015-10-07'});

var params = {
  Entries: [
    {
      Detail: '{ \"key1\": \"value1\", \"key2\": \"value2\" }',
      DetailType: 'appRequestSubmitted',
      Resources: [
        'RESOURCE_ARN',
      ],
      Source: 'com.company.app'
    }
  ]
};

cwevents.putEvents(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutEvents](#) in [AWS SDK for JavaScript API Reference](#).

## CloudWatch Logs examples using SDK for JavaScript V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V2 with Amazon CloudWatch Logs.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2818\)](#)

## Actions

### Create a subscription filter

The following code example shows how to create an Amazon CloudWatch Logs subscription filter.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch Logs service client object.
export const cwlClient = new CloudWatchLogsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {
  PutSubscriptionFilterCommand,
} from "@aws-sdk/client-cloudwatch-logs";
import { cwlClient } from "./libs/cloudWatchLogsClient.js";

// Set the parameters
export const params = {
  destinationArn: "LAMBDA_FUNCTION_ARN", //LAMBDA_FUNCTION_ARN
  filterName: "FILTER_NAME", //FILTER_NAME
  filterPattern: "ERROR",
  logGroupName: "LOG_GROUP", //LOG_GROUP
};

export const run = async () => {
  try {
    const data = await cwlClient.send(new PutSubscriptionFilterCommand(params));
    console.log("Success", data.subscriptionFilters);
    return data; //For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutSubscriptionFilter](#) in [AWS SDK for JavaScript API Reference](#).

#### SDK for JavaScript V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
```

```
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({apiVersion: '2014-03-28'});

var params = {
  destinationArn: 'LAMBDA_FUNCTION_ARN',
  filterName: 'FILTER_NAME',
  filterPattern: 'ERROR',
  logGroupName: 'LOG_GROUP',
};

cwl.putSubscriptionFilter(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutSubscriptionFilter](#) in *AWS SDK for JavaScript API Reference*.

## Delete a subscription filter

The following code example shows how to delete an Amazon CloudWatch Logs subscription filter.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch Logs service client object.
export const cwlClient = new CloudWatchLogsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {
  DeleteSubscriptionFilterCommand
} from "@aws-sdk/client-cloudwatch-logs";
import { cwlClient } from "./libs/cloudWatchLogsClient.js";

// Set the parameters
export const params = {
  filterName: "FILTER", //FILTER
  logGroupName: "LOG_GROUP", //LOG_GROUP
};

export const run = async () => {
```

```
try {
  const data = await cwlClient.send(
    new DeleteSubscriptionFilterCommand(params)
  );
  console.log(
    "Success, subscription filter deleted",
    data
  );
  return data; //For unit tests.
} catch (err) {
  console.log("Error", err);
}
};

// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteSubscriptionFilter](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({apiVersion: '2014-03-28'});

var params = {
  filterName: 'FILTER',
  logGroupName: 'LOG_GROUP'
};

cwl.deleteSubscriptionFilter(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteSubscriptionFilter](#) in [AWS SDK for JavaScript API Reference](#).

## Describe existing subscription filters

The following code example shows how to describe Amazon CloudWatch Logs existing subscription filters.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch Logs service client object.
export const cwlClient = new CloudWatchLogsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { DescribeSubscriptionFiltersCommand } from "@aws-sdk/client-cloudwatch-logs";
import { cwlClient } from "./libs/cloudWatchLogsClient.js";

// Set the parameters
export const params = {
  logGroupName: "GROUP_NAME", //GROUP_NAME
  limit: 5
};

export const run = async () => {
  try {
    const data = await cwlClient.send(
      new DescribeSubscriptionFiltersCommand(params)
    );
    console.log("Success", data.subscriptionFilters);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeSubscriptionFilters](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({apiVersion: '2014-03-28'});

var params = {
  logGroupName: 'GROUP_NAME',
  limit: 5
};

cwl.describeSubscriptionFilters(params, function(err, data) {
  if (err) {
```

```
        console.log("Error", err);
    } else {
        console.log("Success", data.subscriptionFilters);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeSubscriptionFilters](#) in [AWS SDK for JavaScript API Reference](#).

## DynamoDB examples using SDK for JavaScript V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V2 with Amazon DynamoDB.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2822\)](#)

## Actions

### Create a table

The following code example shows how to create a DynamoDB table.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon DynamoDB service client object.
const ddbClient = new DynamoDBClient({ region: REGION });
export { ddbClient };
```

Create the table.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateTableCommand } from "@aws-sdk/client-dynamodb";
import { ddbClient } from "./libs/ddbClient.js";

// Set the parameters
export const params = {
    AttributeDefinitions: [
        {
            AttributeName: "Season", //ATTRIBUTE_NAME_1
```

```
        AttributeType: "N", //ATTRIBUTE_TYPE
    },
    {
        AttributeName: "Episode", //ATTRIBUTE_NAME_2
        AttributeType: "N", //ATTRIBUTE_TYPE
    },
],
KeySchema: [
    {
        AttributeName: "Season", //ATTRIBUTE_NAME_1
        KeyType: "HASH",
    },
    {
        AttributeName: "Episode", //ATTRIBUTE_NAME_2
        KeyType: "RANGE",
    },
],
ProvisionedThroughput: {
    ReadCapacityUnits: 1,
    WriteCapacityUnits: 1,
},
TableName: "TEST_TABLE", //TABLE_NAME
StreamSpecification: {
    StreamEnabled: false,
},
};

export const run = async () => {
    try {
        const data = await ddbClient.send(new CreateTableCommand(params));
        console.log("Table Created", data);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateTable](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
    AttributeDefinitions: [
        {
            AttributeName: 'CUSTOMER_ID',
            AttributeType: 'N'
        },
        {
            AttributeName: 'CUSTOMER_NAME',
            AttributeType: 'S'
        }
    ],
    KeySchema: [
        {
            AttributeName: "CustomerID", //ATTRIBUTE_NAME_1
            KeyType: "HASH"
        },
        {
            AttributeName: "CustomerName", //ATTRIBUTE_NAME_2
            KeyType: "RANGE"
        }
    ],
    ProvisionedThroughput: {
        ReadCapacityUnits: 1,
        WriteCapacityUnits: 1
    }
};
```

```
        AttributeType: 'S'
    },
],
KeySchema: [
    {
        AttributeName: 'CUSTOMER_ID',
        KeyType: 'HASH'
    },
    {
        AttributeName: 'CUSTOMER_NAME',
        KeyType: 'RANGE'
    }
],
ProvisionedThroughput: {
    ReadCapacityUnits: 1,
    WriteCapacityUnits: 1
},
TableName: 'CUSTOMER_LIST',
StreamSpecification: {
    StreamEnabled: false
}
};

// Call DynamoDB to create the table
ddb.createTable(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Table Created", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateTable](#) in [AWS SDK for JavaScript API Reference](#).

## Delete a table

The following code example shows how to delete a DynamoDB table.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Delete the table.

```
// Import required AWS SDK clients and commands for Node.js
import { DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import { ddbClient } from "./libs/ddbClient.js";
```

```
// Set the parameters
export const params = {
  TableName: "CUSTOMER_LIST_NEW",
};

export const run = async () => {
  try {
    const data = await ddbClient.send(new DeleteTableCommand(params));
    console.log("Success, table deleted", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [DeleteTable](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
  TableName: process.argv[2]
};

// Call DynamoDB to delete the specified table
ddb.deleteTable(params, function(err, data) {
  if (err && err.code === 'ResourceNotFoundException') {
    console.log("Error: Table not found");
  } else if (err && err.code === 'ResourceInUseException') {
    console.log("Error: Table in use");
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteTable](#) in *AWS SDK for JavaScript API Reference*.

## Delete an item from a table

The following code example shows how to delete an item from a DynamoDB table.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.  
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";  
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";  
// Create an Amazon DynamoDB service client object.  
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the document client.

```
// Create a service client module using ES6 syntax.  
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";  
import { ddbClient } from "./ddbClient.js";  
  
const marshallOptions = {  
    // Whether to automatically convert empty strings, blobs, and sets to `null`.  
    convertEmptyValues: false, // false, by default.  
    // Whether to remove undefined values while marshalling.  
    removeUndefinedValues: true, // false, by default.  
    // Whether to convert typeof object to map attribute.  
    convertClassInstanceToMap: false, // false, by default.  
};  
  
const unmarshallOptions = {  
    // Whether to return numbers as a string instead of converting them to native  
    // JavaScript numbers.  
    wrapNumbers: false, // false, by default.  
};  
  
// Create the DynamoDB document client.  
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {  
    marshallOptions,  
    unmarshallOptions,  
});  
  
export { ddbDocClient };
```

Delete an item from a table using the DynamoDB document client.

```
import { DeleteCommand } from "@aws-sdk/lib-dynamodb";  
import { ddbDocClient } from "../../libs/ddbDocClient.js";  
  
// Set the parameters.  
export const params = {  
    TableName: "TABLE_NAME",  
    Key: {  
        primaryKey: "VALUE_1",  
        sortKey: "VALUE_2",  
    },  
};  
  
export const deleteItem = async () => {  
    try {  
        await ddbDocClient.send(new DeleteCommand(params));  
        console.log("Success - item deleted");  
    } catch (err) {  
        console.log("Error", err);  
    }  
};
```

```
deleteItem();
```

Delete an item from a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];

export const run = async (tableName, movieYear1, movieTitle1) => {
  const params = {
    Statement: "DELETE FROM " + tableName + " where title=? and year=?",
    Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
  };
  try {
    await ddbDocClient.send(new ExecuteStatementCommand(params));
    console.log("Success. Item deleted.");
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.error(err);
  }
};
run(tableName, movieYear1, movieTitle1);
```

Delete an item from a table by batch using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[5];
const movieTitle2 = process.argv[6];

export const run = async (
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2
) => {
  try {
    const params = {
      Statements: [
        {
          Statement: "DELETE FROM " + tableName + " where year=? and title=?",
          Parameters: [{ N: movieYear1 }, { S: movieTitle1 }],
        },
        {
          Statement: "DELETE FROM " + tableName + " where year=? and title=?",
          Parameters: [{ N: movieYear2 }, { S: movieTitle2 }],
        },
      ],
    };
    const data = await ddbDocClient.send(
```

```
        new BatchExecuteStatementCommand(params)
    );
    console.log("Success. Items deleted.", data);
    return "Run successfully"; // For unit tests.
} catch (err) {
    console.error(err);
}
};

run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteItem](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an item from a table.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
    TableName: 'TABLE',
    Key: {
        'KEY_NAME': {N: 'VALUE'}
    }
};

// Call DynamoDB to delete the item from the table
ddb.deleteItem(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

Delete an item from a table using the DynamoDB document client.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({apiVersion: '2012-08-10'});

var params = {
    Key: {
        'HASH_KEY': VALUE
    },
    TableName: 'TABLE'
};
```

```
docClient.delete(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteItem](#) in *AWS SDK for JavaScript API Reference*.

## Get a batch of items

The following code example shows how to get a batch of DynamoDB items.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon DynamoDB service client object.
const ddbClient = new DynamoDBClient({ region: REGION });
export { ddbClient };
```

Get the items.

```
// Import required AWS SDK clients and commands for Node.js
import { BatchGetItemCommand } from "@aws-sdk/client-dynamodb";
import { ddbClient } from "./libs/ddbClient.js";

// Set the parameters
export const params = {
  RequestItems: {
    TABLE_NAME: {
      Keys: [
        {
          KEY_NAME_1: { N: "KEY_VALUE" },
          KEY_NAME_2: { N: "KEY_VALUE" },
          KEY_NAME_3: { N: "KEY_VALUE" },
        },
      ],
      ProjectionExpression: "ATTRIBUTE_NAME",
    },
  },
};

export const run = async () => {
  try {
    const data = await ddbClient.send(new BatchGetItemCommand(params));
    console.log("Success, items retrieved", data);
    return data;
  } catch (err) {
```

```
        console.log("Error", err);
    }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [BatchGetItem](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
  RequestItems: [
    {
      'TABLE_NAME': {
        Keys: [
          {'KEY_NAME': {N: 'KEY_VALUE_1'}},
          {'KEY_NAME': {N: 'KEY_VALUE_2'}},
          {'KEY_NAME': {N: 'KEY_VALUE_3'}}
        ],
        ProjectionExpression: 'KEY_NAME, ATTRIBUTE'
      }
    }
  ];
};

ddb.batchGetItem(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    data.Responses.TABLE_NAME.forEach(function(element, index, array) {
      console.log(element);
    });
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [BatchGetItem](#) in [AWS SDK for JavaScript API Reference](#).

## Get an item from a table

The following code example shows how to get an item from a DynamoDB table.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.  
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";  
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";  
// Create an Amazon DynamoDB service client object.  
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.  
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";  
import { ddbClient } from "./ddbClient.js";  
  
const marshallOptions = {  
    // Whether to automatically convert empty strings, blobs, and sets to `null`.  
    convertEmptyValues: false, // false, by default.  
    // Whether to remove undefined values while marshalling.  
    removeUndefinedValues: true, // false, by default.  
    // Whether to convert typeof object to map attribute.  
    convertClassInstanceToMap: false, // false, by default.  
};  
  
const unmarshallOptions = {  
    // Whether to return numbers as a string instead of converting them to native  
    // JavaScript numbers.  
    wrapNumbers: false, // false, by default.  
};  
  
// Create the DynamoDB document client.  
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {  
    marshallOptions,  
    unmarshallOptions,  
});  
  
export { ddbDocClient };
```

Get an item from a table.

```
import { GetCommand } from "@aws-sdk/lib-dynamodb";  
import { ddbDocClient } from "../libs/ddbDocClient.js";  
  
// Set the parameters.  
export const params = {  
    TableName: "TABLE_NAME",  
    Key: {  
        primaryKey: "VALUE_1",  
        sortKey: "VALUE_2",  
    },  
};  
  
export const getItem = async () => {  
    try {  
        const data = await ddbDocClient.send(new GetCommand(params));  
        console.log("Success :", data.Item);  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
getItem();
```

Get an item from a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient";

const tableName = process.argv[2];
const movieTitle1 = process.argv[3];

export const run = async (tableName, movieTitle1) => {
  const params = {
    Statement: "SELECT * FROM " + tableName + " where title=?",
    Parameters: [{ S: movieTitle1 }],
  };
  try {
    const data = await ddbDocClient.send(new ExecuteStatementCommand(params));
    for (let i = 0; i < data.Items.length; i++) {
      console.log(
        "Success. The query return the following data. Item " + i,
        data.Items[i].year,
        data.Items[i].title,
        data.Items[i].info
      );
    }
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.error(err);
  }
};
run(tableName, movieTitle1);
```

Get items by batch from a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[5];
const movieTitle2 = process.argv[6];

export const run = async (
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2
) => {
  const params = {
    Statements: [
      {
        Statement: "SELECT * FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
      },
      {
        Statement: "SELECT * FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
      },
    ],
  };
  const data = await ddbDocClient.send(new BatchExecuteStatementCommand(params));
  return data;
};
```

```
    Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
  ],
};

try {
  const data = await ddbDocClient.send(
    new BatchExecuteStatementCommand(params)
  );
  for (let i = 0; i < data.Responses.length; i++) {
    console.log(data.Responses[i].Item.year);
    console.log(data.Responses[i].Item.title);
  }
  return "Run successfully"; // For unit tests.
} catch (err) {
  console.error(err);
}
};

run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

- For API details, see [GetItem](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get an item from a table.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
  TableName: 'TABLE',
  Key: {
    'KEY_NAME': {N: '001'}
  },
  ProjectionExpression: 'ATTRIBUTE_NAME'
};

// Call DynamoDB to read the item from the table
ddb.getItem(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Item);
  }
});
```

Get an item from a table using the DynamoDB document client.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});
```

```
// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({apiVersion: '2012-08-10'});

var params = {
  TableName: 'EPISODES_TABLE',
  Key: {'KEY_NAME': VALUE}
};

docClient.get(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Item);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetItem](#) in [AWS SDK for JavaScript API Reference](#).

## Get information about a table

The following code example shows how to get information about a DynamoDB table.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon DynamoDB service client object.
const ddbClient = new DynamoDBClient({ region: REGION });
export { ddbClient };
```

Describe the table.

```
// Import required AWS SDK clients and commands for Node.js
import { DescribeTableCommand } from "@aws-sdk/client-dynamodb";
import { ddbClient } from "./libs/ddbClient.js";

// Set the parameters
export const params = { TableName: "TABLE_NAME" }; //TABLE_NAME

export const run = async () => {
  try {
    const data = await ddbClient.send(new DescribeTableCommand(params));
    console.log("Success", data);
    // console.log("Success", data.Table.KeySchema);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeTable](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
  TableName: process.argv[2]
};

// Call DynamoDB to retrieve the selected table descriptions
ddb.describeTable(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Table.KeySchema);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeTable](#) in [AWS SDK for JavaScript API Reference](#).

### List tables

The following code example shows how to list DynamoDB tables.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon DynamoDB service client object.
const ddbClient = new DynamoDBClient({ region: REGION });
export { ddbClient };
```

List the tables.

```
// Import required AWS SDK clients and commands for Node.js
import { ListTablesCommand } from "@aws-sdk/client-dynamodb";
import { ddbClient } from "./libs/ddbClient.js";

export const run = async () => {
  try {
    const data = await ddbClient.send(new ListTablesCommand({}));
    console.log(data);
    // console.log(data.TableNames.join("\n"));
    return data;
  } catch (err) {
    console.error(err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListTables](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

// Call DynamoDB to retrieve the list of tables
ddb.listTables({Limit: 10}, function(err, data) {
  if (err) {
    console.log("Error", err.code);
  } else {
    console.log("Table names are ", data.TableNames);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListTables](#) in [AWS SDK for JavaScript API Reference](#).

### Put an item in a table

The following code example shows how to put an item in a DynamoDB table.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
```

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: true, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {
    marshallOptions,
    unmarshallOptions,
});

export { ddbDocClient };
```

Put an item in a table using the DynamoDB document client.

```
import { PutCommand } from "@aws-sdk/lib-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

export const putItem = async () => {
    // Set the parameters.
    export const params = {
        TableName: "TABLE_NAME",
        Item: {
            primaryKey: "VALUE_1",
            sortKey: "VALUE_2",
        },
    };
    try {
        const data = await ddbDocClient.send(new PutCommand(params));
        console.log("Success - item added or updated", data);
    } catch (err) {
        console.log("Error", err.stack);
    }
};
putItem();
```

Put an item in a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];

export const run = async (tableName, movieTitle1, movieYear1) => {
    const params = {
        Statement: "INSERT INTO " + tableName + " value {'title':?, 'year':?}",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
    };
    try {
        await ddbDocClient.send(new ExecuteStatementCommand(params));
        console.log("Success. Item added.");
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.error(err);
    }
};
run(tableName, movieTitle1, movieYear1);
```

Put items by batch in a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[5];
const movieTitle2 = process.argv[6];

export const run = async (
    tableName,
    movieYear1,
    movieTitle1,
    movieYear2,
    movieTitle2
) => {
    const params = {
        Statements: [
            {
                Statement:
                    "INSERT INTO " + tableName + " value {'title':?, 'year':?}",
                Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
            },
            {
                Statement:
                    "INSERT INTO " + tableName + " value {'title':?, 'year':?}",
                Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
            },
        ],
    };
    try {
        await ddbDocClient.send(
            new BatchExecuteStatementCommand(params)
        );
        console.log("Success. Items added.");
    } catch (err) {
        console.error(err);
    }
};
```

```
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.error(err);
    }
};

run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

- For API details, see [PutItem](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Put an item in a table.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
    TableName: 'CUSTOMER_LIST',
    Item: {
        'CUSTOMER_ID' : {N: '001'},
        'CUSTOMER_NAME' : {S: 'Richard Roe'}
    }
};

// Call DynamoDB to add the item to the table
ddb.putItem(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

Put an item in a table using the DynamoDB document client.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({apiVersion: '2012-08-10'});

var params = {
    TableName: 'TABLE',
    Item: {
        'HASHKEY': VALUE,
        'ATTRIBUTE_1': 'STRING_VALUE',
        'ATTRIBUTE_2': VALUE_2
    }
};
```

```
docClient.put(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutItem](#) in *AWS SDK for JavaScript API Reference*.

## Query a table

The following code example shows how to query a DynamoDB table.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon DynamoDB service client object.
const ddbClient = new DynamoDBClient({ region: REGION });
export { ddbClient };
```

Query the table.

```
// Import required AWS SDK clients and commands for Node.js
import { QueryCommand } from "@aws-sdk/client-dynamodb";
import { ddbClient } from "./libs/ddbClient.js";

// Set the parameters
export const params = {
  KeyConditionExpression: "Season = :s and Episode > :e",
  FilterExpression: "contains (Subtitle, :topic)",
  ExpressionAttributeValues: {
    ":s": { N: "1" },
    ":e": { N: "2" },
    ":topic": { S: "SubTitle" },
  },
  ProjectionExpression: "Episode, Title, Subtitle",
  TableName: "EPISODES_TABLE",
};

export const run = async () => {
  try {
    const data = await ddbClient.send(new QueryCommand(params));
    data.Items.forEach(function (element) {
      console.log(element.Title.S + " (" + element.Subtitle.S + ")");
    });
    return data;
  } catch (err) {
```

```
        console.error(err);
    }
};

run();
```

Create the client for the DynamoDB document client.

```
// Create service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: false, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

const translateConfig = { marshallOptions, unmarshallOptions };

// Create the DynamoDB Document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, translateConfig);

export { ddbDocClient };
```

Query the table using the DynamoDB document client.

```
import { QueryCommand } from "@aws-sdk/lib-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

// Set the parameters.
export const params = {
    ExpressionAttributeNames: { "#r": "rank", "#y": "year" },
    ProjectionExpression: "#r, #y, title",
    TableName: "TABLE_NAME",
    ExpressionAttributeValues: {
        ":t": "MOVIE_NAME",
        ":y": "MOVIE_YEAR",
        ":r": "MOVIE_RANK",
    },
    KeyConditionExpression: "title = :t and #y = :y",
    FilterExpression: "info.#r = :r",
};

export const queryTable = async () => {
    try {
        const data = await ddbDocClient.send(new QueryCommand(params));
        for (let i = 0; i < data.Items.length; i++) {
            console.log(
                "Success. Items with rank of " +
                "MOVIE_RANK" +
                " include\n" +
                "Year = " +
            );
        }
    } catch (err) {
        console.error("Error querying table: ", err);
    }
};
```

```
        data.Items[i].year +
        " Title = " +
        data.Items[i].title
    );
}
} catch (err) {
    console.log("Error", err);
}
};

queryTable();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Query in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({apiVersion: '2012-08-10'});

var params = {
    ExpressionAttributeValues: {
        ':s': 2,
        ':e': 9,
        ':topic': 'PHRASE'
    },
    KeyConditionExpression: 'Season = :s and Episode > :e',
    FilterExpression: 'contains (Subtitle, :topic)',
    TableName: 'EPISODES_TABLE'
};

docClient.query(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data.Items);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Query in AWS SDK for JavaScript API Reference](#).

## Scan a table

The following code example shows how to scan a DynamoDB table.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.  
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";  
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";  
// Create an Amazon DynamoDB service client object.  
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.  
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";  
import { ddbClient } from "./ddbClient.js";  
  
const marshallOptions = {  
    // Whether to automatically convert empty strings, blobs, and sets to `null`.  
    convertEmptyValues: false, // false, by default.  
    // Whether to remove undefined values while marshalling.  
    removeUndefinedValues: true, // false, by default.  
    // Whether to convert typeof object to map attribute.  
    convertClassInstanceToMap: false, // false, by default.  
};  
  
const unmarshallOptions = {  
    // Whether to return numbers as a string instead of converting them to native  
    // JavaScript numbers.  
    wrapNumbers: false, // false, by default.  
};  
  
// Create the DynamoDB document client.  
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {  
    marshallOptions,  
    unmarshallOptions,  
});  
  
export { ddbDocClient };
```

Scan a table using the DynamoDB document client.

```
// Import required AWS SDK clients and commands for Node.js.  
import { ddbDocClient } from "../../libs/ddbDocClient.js";  
import { ScanCommand } from "@aws-sdk/lib-dynamodb";  
// Set the parameters.  
export const params = {  
    TableName: "TABLE_NAME",  
    ProjectionExpression: "#r, #y, title",  
    ExpressionAttributeNames: { "#r": "rank", "#y": "year" },  
    FilterExpression: "title = :t and #y = :y and info.#r = :r",  
    ExpressionAttributeValues: {  
        ":r": "MOVIE_RANK",  
        ":y": "MOVIE_YEAR",  
        ":t": "MOVIE_NAME",  
    },  
};  
  
export const scanTable = async () => {  
    try {  
        const data = await ddbDocClient.send(new ScanCommand(params));  
    }
```

```
        console.log("success", data.Items);
    } catch (err) {
        console.log("Error", err);
    }
};

scanTable();
```

- For API details, see [Scan](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js.
var AWS = require("aws-sdk");
// Set the AWS Region.
AWS.config.update({ region: "REGION" });

// Create DynamoDB service object.
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

const params = {
    // Specify which items in the results are returned.
    FilterExpression: "Subtitle = :topic AND Season = :s AND Episode = :e",
    // Define the expression attribute value, which are substitutes for the values you
    // want to compare.
    ExpressionAttributeValues: {
        ":topic": {S: "SubTitle2"},
        ":s": {N: 1},
        ":e": {N: 2},
    },
    // Set the projection expression, which are the attributes that you want.
    ProjectionExpression: "Season, Episode, Title, Subtitle",
    TableName: "EPISODES_TABLE",
};

ddb.scan(params, function (err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
        data.Items.forEach(function (element, index, array) {
            console.log(
                "printing",
                element.Title.S + " (" + element.Subtitle.S + ")"
            );
        });
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Scan](#) in *AWS SDK for JavaScript API Reference*.

## Write a batch of items

The following code example shows how to write a batch of DynamoDB items.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../../libs/utils/util-aws-sdk.js";
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: true, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {
    marshallOptions,
    unmarshallOptions,
});

export { ddbDocClient };
```

Add the items to the table.

```
import fs from "fs";
import * as R from "ramda";
import { ddbDocClient } from "../../../libs/ddbDocClient.js";
import { BatchWriteCommand } from "@aws-sdk/lib-dynamodb";

export const writeData = async () => {
    // Before you run this example, download 'movies.json' from https://
    // docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Js.02.html,
    // and put it in the same folder as the example.
    // Get the movie data parse to convert into a JSON object.
    const allMovies = JSON.parse(fs.readFileSync("moviedata.json", "utf8"));
    // Split the table into segments of 25.
    const dataSegments = R.splitEvery(25, allMovies);
```

```
const TABLE_NAME = "TABLE_NAME"
try {
    // Loop batch write operation 10 times to upload 250 items.
    for (let i = 0; i < 10; i++) {
        const segment = dataSegments[i];
        for (let j = 0; j < 25; j++) {
            const params = {
                RequestItems: {
                    [TABLE_NAME]: [
                        {
                            PutRequest: {
                                Item: {
                                    year: segment[j].year,
                                    title: segment[j].title,
                                    info: segment[j].info,
                                },
                            },
                        ],
                    ],
                };
               ddbDocClient.send(new BatchWriteCommand(params));
            }
            console.log("Success, table updated.");
        }
    } catch (error) {
        console.log("Error", error);
    }
};
writeData();
```

- For API details, see [BatchWriteItem in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
    RequestItems: {
        "TABLE_NAME": [
            {
                PutRequest: {
                    Item: {
                        "KEY": { "N": "KEY_VALUE" },
                        "ATTRIBUTE_1": { "S": "ATTRIBUTE_1_VALUE" },
                        "ATTRIBUTE_2": { "N": "ATTRIBUTE_2_VALUE" }
                    }
                }
            },
            {
                PutRequest: {
                    Item: {
```

```
        "KEY": { "N": "KEY_VALUE" },
        "ATTRIBUTE_1": { "S": "ATTRIBUTE_1_VALUE" },
        "ATTRIBUTE_2": { "N": "ATTRIBUTE_2_VALUE" }
    }
}
];
};

ddb.batchWriteItem(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [BatchWriteItem in AWS SDK for JavaScript API Reference](#).

## EventBridge examples using SDK for JavaScript V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V2 with Amazon EventBridge.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2847\)](#)

## Actions

### Add a Lambda function target

The following code example shows how to add an AWS Lambda function target to an Amazon EventBridge event.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { EventBridgeClient } from "@aws-sdk/client-eventbridge";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon EventBridge service client object.
export const ebClient = new EventBridgeClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js.
import { PutTargetsCommand } from "@aws-sdk/client-eventbridge";
import { ebClient } from "./libs/eventBridgeClient.js";

// Set the parameters.
export const params = {
  Rule: "DEMO_EVENT",
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN", //LAMBDA_FUNCTION_ARN
      Id: "myCloudWatchEventsTarget",
    },
  ],
};

export const run = async () => {
  try {
    const data = await ebClient.send(new PutTargetsCommand(params));
    console.log("Success, target added; requestID: ", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For API details, see [PutTargets](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({apiVersion: '2015-10-07'});

var params = {
  Rule: 'DEMO_EVENT',
  Targets: [
    {
      Arn: 'LAMBDA_FUNCTION_ARN',
      Id: 'myEventBridgeTarget',
    }
  ]
};

ebevents.putTargets(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For API details, see [PutTargets](#) in *AWS SDK for JavaScript API Reference*.

## Create a scheduled rule

The following code example shows how to create an Amazon EventBridge scheduled rule.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { EventBridgeClient } from "@aws-sdk/client-eventbridge";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon EventBridge service client object.
export const ebClient = new EventBridgeClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js.
import { PutRuleCommand } from "@aws-sdk/client-eventbridge";
import { ebClient } from "./libs/eventBridgeClient.js";

// Set the parameters.
export const params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN", //IAM_ROLE_ARN
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

export const run = async () => {
  try {
    const data = await ebClient.send(new PutRuleCommand(params));
    console.log("Success, scheduled rule created; Rule ARN:", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For API details, see [PutRule](#) in *AWS SDK for JavaScript API Reference*.

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
```

```
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({apiVersion: '2015-10-07'});

var params = {
  Name: 'DEMO_EVENT',
  RoleArn: 'IAM_ROLE_ARN',
  ScheduleExpression: 'rate(5 minutes)',
  State: 'ENABLED'
};

ebevents.putRule(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.RuleArn);
  }
});
```

- For API details, see [PutRule](#) in *AWS SDK for JavaScript API Reference*.

## Send events

The following code example shows how to send Amazon EventBridge events.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { EventBridgeClient } from "@aws-sdk/client-eventbridge";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon EventBridge service client object.
export const ebClient = new EventBridgeClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js.
import { PutEventsCommand } from "@aws-sdk/client-eventbridge";
import { ebClient } from "./libs/eventBridgeClient.js";

// Set the parameters.
export const params = {
  Entries: [
    {
      Detail: '{ "key1": "value1", "key2": "value2" }',
      DetailType: "appRequestSubmitted",
      Resources: [
        "RESOURCE_ARN", //RESOURCE_ARN
      ],
      Source: "com.company.app",
```

```
        },
    ],
};

export const run = async () => {
    try {
        const data = await ebClient.send(new PutEventsCommand(params));
        console.log("Success, event sent; requestID:", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
// Uncomment this line to run execution within this file.
// run();
```

- For API details, see [PutEvents](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({apiVersion: '2015-10-07'});

var params = {
    Entries: [
        {
            Detail: '{ \"key1\": \"value1\", \"key2\": \"value2\" }',
            DetailType: 'appRequestSubmitted',
            Resources: [
                'RESOURCE_ARN',
            ],
            Source: 'com.company.app'
        }
    ]
};

ebevents.putEvents(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data.Entries);
    }
});
```

- For API details, see [PutEvents](#) in *AWS SDK for JavaScript API Reference*.

## IAM examples using SDK for JavaScript V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V2 with AWS Identity and Access Management.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2852\)](#)

## Actions

### Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Attach the policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import {
  ListAttachedRolePoliciesCommand,
  AttachRolePolicyCommand,
} from "@aws-sdk/client-iam";

// Set the parameters.
const ROLENAMES = ["ROLE_NAME"];
const paramsRoleList = { RoleName: ROLENAMES }; //ROLE_NAME
export const params = {
  PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
  RoleName: ROLENAMES,
};
export const run = async () => {
  try {
    const data = await iamClient.send(
      new ListAttachedRolePoliciesCommand(paramsRoleList)
    );
    const myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (_val, index) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        console.log(
          "AmazonDynamoDBFullAccess is already attached to this role."
        );
        process.exit();
      }
    });
  } catch (err) {
    console.error(`Error: ${err}`);
  }
};
```

```
    const data = await iamClient.send(new AttachRolePolicyCommand(params));
    console.log("Role attached successfully");
    return data;
} catch (err) {
    console.log("Error", err);
}
} catch (err) {
    console.log("Error", err);
}
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [AttachRolePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var paramsRoleList = {
    RoleName: process.argv[2]
};

iam.listAttachedRolePolicies(paramsRoleList, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        var myRolePolicies = data.AttachedPolicies;
        myRolePolicies.forEach(function (val, index, array) {
            if (myRolePolicies[index].PolicyName === 'AmazonDynamoDBFullAccess') {
                console.log("AmazonDynamoDBFullAccess is already attached to this role.");
                process.exit();
            }
        });
        var params = {
            PolicyArn: 'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess',
            RoleName: process.argv[2]
        };
        iam.attachRolePolicy(params, function(err, data) {
            if (err) {
                console.log("Unable to attach policy to role", err);
            } else {
                console.log("Role attached successfully");
            }
        });
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [AttachRolePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## Create a policy

The following code example shows how to create an IAM policy.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Create the policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { CreatePolicyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const myManagedPolicy = {
  Version: "2012-10-17",
  Statement: [
    {
      Effect: "Allow",
      Action: "logs:CreateLogGroup",
      Resource: "RESOURCE_ARN", // RESOURCE_ARN
    },
    {
      Effect: "Allow",
      Action: [
        "dynamodb>DeleteItem",
        "dynamodb>GetItem",
        "dynamodb>PutItem",
        "dynamodb>Scan",
        "dynamodb>UpdateItem",
      ],
      Resource: "DYNAMODB_POLICY_NAME", // DYNAMODB_POLICY_NAME; For example,
      "myDynamoDBName".
    },
  ],
};

export const params = {
  PolicyDocument: JSON.stringify(myManagedPolicy),
  PolicyName: "IAM_POLICY_NAME",
};

export const run = async () => {
  try {
    const data = await iamClient.send(new CreatePolicyCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreatePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var myManagedPolicy = {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs>CreateLogGroup",
            "Resource": "RESOURCE_ARN"
        },
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb>DeleteItem",
                "dynamodb>GetItem",
                "dynamodb>PutItem",
                "dynamodb>Scan",
                "dynamodb>UpdateItem"
            ],
            "Resource": "RESOURCE_ARN"
        }
    ]
};

var params = {
    PolicyDocument: JSON.stringify(myManagedPolicy),
    PolicyName: 'myDynamoDBPolicy',
};

iam.createPolicy(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreatePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## Create a user

The following code example shows how to create an IAM user.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Create the user.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { GetUserCommand, CreateUserCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { UserName: "USER_NAME" }; //USER_NAME

export const run = async () => {
  try {
    const data = await iamClient.send(new GetUserCommand(params));
    console.log(
      "User " + "USER_NAME" + " already exists",
      data.User.UserId
    );
    return data;
  } catch (err) {
    try {
      const results = await iamClient.send(new CreateUserCommand(params));
      console.log("Success", results);
      return results;
    } catch (err) {
      console.log("Error", err);
    }
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateUser](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});
```

```
var params = {
  UserName: process.argv[2]
};

iam.getUser(params, function(err, data) {
  if (err && err.code === 'NoSuchEntity') {
    iam.createUser(params, function(err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  } else {
    console.log("User " + process.argv[2] + " already exists", data.User.UserId);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateUser](#) in [AWS SDK for JavaScript API Reference](#).

## Create an access key

The following code example shows how to create an IAM access key.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Create the access key.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { CreateAccessKeyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {UserName: "IAM_USER_NAME"}; //IAM_USER_NAME

export const run = async () => {
  try {
    const data = await iamClient.send(new CreateAccessKeyCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
```

```
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateAccessKey in AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.createAccessKey({UserName: 'IAM_USER_NAME'}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.AccessKey);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateAccessKey in AWS SDK for JavaScript API Reference](#).

### Create an alias for an account

The following code example shows how to create an alias for an IAM account.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Create the account alias.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { CreateAccountAliasCommand } from "@aws-sdk/client-iam";

// Set the parameters.
```

```
export const params = { AccountAlias: "ACCOUNT_ALIAS" }; //ACCOUNT_ALIAS

export const run = async () => {
  try {
    const data = await iamClient.send(new CreateAccountAliasCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateAccountAlias](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.createAccountAlias({AccountAlias: process.argv[2]}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateAccountAlias](#) in [AWS SDK for JavaScript API Reference](#).

### Delete a server certificate

The following code example shows how to delete an IAM server certificate.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
```

```
export { iamClient };
```

Delete a server certificate.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { DeleteServerCertificateCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { ServerCertificateName: "CERTIFICATE_NAME" }; // CERTIFICATE_NAME

export const run = async () => {
  try {
    const data = await iamClient.send(
      new DeleteServerCertificateCommand(params)
    );
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteServerCertificate](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.deleteServerCertificate({ServerCertificateName: 'CERTIFICATE_NAME'}, function(err,
  data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteServerCertificate](#) in [AWS SDK for JavaScript API Reference](#).

## Delete a user

The following code example shows how to delete an IAM user.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Delete the user.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { DeleteUserCommand, GetUserCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { UserName: "USER_NAME" }; //USER_NAME

export const run = async () => {
  try {
    const data = await iamClient.send(new GetUserCommand(params));
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteUser](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  UserName: process.argv[2]
```

```
};

iam.getUser(params, function(err, data) {
  if (err && err.code === 'NoSuchEntity') {
    console.log("User " + process.argv[2] + " does not exist.");
  } else {
    iam.deleteUser(params, function(err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteUser](#) in [AWS SDK for JavaScript API Reference](#).

## Delete an access key

The following code example shows how to delete an IAM access key.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Delete the access key.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { DeleteAccessKeyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  AccessKeyId: "ACCESS_KEY_ID", // ACCESS_KEY_ID
  UserName: "USER_NAME", // USER_NAME
};

export const run = async () => {
  try {
    const data = await iamClient.send(new DeleteAccessKeyCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAccessKey](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  AccessKeyId: 'ACCESS_KEY_ID',
  UserName: 'USER_NAME'
};

iam.deleteAccessKey(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAccessKey](#) in [AWS SDK for JavaScript API Reference](#).

## Delete an account alias

The following code example shows how to delete an IAM account alias.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Delete the account alias.

```
// Import required AWS SDK clients and commands for Node.js.
```

```
import { iamClient } from "./libs/iamClient.js";
import { DeleteAccountAliasCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { AccountAlias: "ALIAS" }; // ALIAS

export const run = async () => {
  try {
    const data = await iamClient.send(new DeleteAccountAliasCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAccountAlias](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.deleteAccountAlias({AccountAlias: process.argv[2]}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAccountAlias](#) in [AWS SDK for JavaScript API Reference](#).

### Detach a policy from a role

The following code example shows how to detach an IAM policy from a role.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
```

```
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

Detach the policy.

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "./libs/iamClient.js";  
import {  
    ListAttachedRolePoliciesCommand,  
    DetachRolePolicyCommand,  
} from "@aws-sdk/client-iam";  
  
// Set the parameters.  
export const params = { RoleName: "ROLE_NAME" }; //ROLE_NAME  
  
export const run = async () => {  
    try {  
        const data = await iamClient.send(  
            new ListAttachedRolePoliciesCommand(params)  
        );  
        const myRolePolicies = data.AttachedPolicies;  
        myRolePolicies.forEach(function (_val, index) {  
            if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {  
                try {  
                    await iamClient.send(  
                        new DetachRolePolicyCommand(paramsRoleList)  
                    );  
                    console.log("Policy detached from role successfully");  
                    process.exit();  
                } catch (err) {  
                    console.log("Unable to detach policy from role", err);  
                }  
            } else {  
            }  
        });  
        return data;  
    } catch (err) {  
        console.log("User " + "USER_NAME" + " does not exist.");  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DetachRolePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create the IAM service object  
var iam = new AWS.IAM({apiVersion: '2010-05-08'});
```

```
var paramsRoleList = {
  RoleName: process.argv[2]
};

iam.listAttachedRolePolicies(paramsRoleList, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === 'AmazonDynamoDBFullAccess') {
        var params = {
          PolicyArn: 'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess',
          RoleName: process.argv[2]
        };
        iam.detachRolePolicy(params, function(err, data) {
          if (err) {
            console.log("Unable to detach policy from role", err);
          } else {
            console.log("Policy detached from role successfully");
            process.exit();
          }
        });
      }
    });
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DetachRolePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## Get a policy

The following code example shows how to get an IAM policy.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Get the policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { GetPolicyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const params = {
```

```
    PolicyArn: "POLICY_ARN" /* required */,  
};  
  
const run = async () => {  
  try {  
    const data = await iamClient.send(new GetPolicyCommand(params));  
    console.log("Success", data.Policy);  
  } catch (err) {  
    console.log("Error", err);  
  }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetPolicy](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create the IAM service object  
var iam = new AWS.IAM({apiVersion: '2010-05-08'});  
  
var params = {  
  PolicyArn: 'arn:aws:iam::aws:policy/AWSLambdaExecute'  
};  
  
iam.getPolicy(params, function(err, data) {  
  if (err) {  
    console.log("Error", err);  
  } else {  
    console.log("Success", data.Policy.Description);  
  }  
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetPolicy](#) in [AWS SDK for JavaScript API Reference](#).

### Get a server certificate

The following code example shows how to get an IAM server certificate.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
```

```
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

Get a server certificate.

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "./libs/iamClient.js";  
import { GetServerCertificateCommand } from "@aws-sdk/client-iam";  
  
// Set the parameters.  
export const params = { ServerCertificateName: "CERTIFICATE_NAME" }; //CERTIFICATE_NAME  
  
export const run = async () => {  
    try {  
        const data = await iamClient.send(new GetServerCertificateCommand(params));  
        console.log("Success", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetServerCertificate](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create the IAM service object  
var iam = new AWS.IAM({apiVersion: '2010-05-08'});  
  
iam.getServerCertificate({ServerCertificateName: 'CERTIFICATE_NAME'}, function(err, data) {  
    if (err) {  
        console.log("Error", err);  
    } else {  
        console.log("Success", data);  
    }  
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetServerCertificate](#) in [AWS SDK for JavaScript API Reference](#).

## Get data about the last use of an access key

The following code example shows how to get data about the last use of an IAM access key.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Get the access key.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { GetAccessKeyLastUsedCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { AccessKeyId: "ACCESS_KEY_ID" }; //ACCESS_KEY_ID

export const run = async () => {
  try {
    const data = await iamClient.send(new GetAccessKeyLastUsedCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetAccessKeyLastUsed](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.getAccessKeyLastUsed({AccessKeyId: 'ACCESS_KEY_ID'}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.AccessKeyLastUsed);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetAccessKeyLastUsed](#) in [AWS SDK for JavaScript API Reference](#).

### List a user's access keys

The following code example shows how to list a user's IAM access keys.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the access keys.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { ListAccessKeysCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  MaxItems: 5,
  UserName: "IAM_USER_NAME", //IAM_USER_NAME
};

export const run = async () => {
  try {
    const data = await iamClient.send(new ListAccessKeysCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListAccessKeys](#) in [AWS SDK for JavaScript API Reference](#).

#### SDK for JavaScript V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
```

```
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  MaxItems: 5,
  UserName: 'IAM_USER_NAME'
};

iam.listAccessKeys(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListAccessKeys](#) in [AWS SDK for JavaScript API Reference](#).

## List account aliases

The following code example shows how to list IAM account aliases.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the account aliases.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { ListAccountAliasesCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { MaxItems: 5 };

export const run = async () => {
  try {
    const data = await iamClient.send(new ListAccountAliasesCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
}
```

```
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListAccountAliases](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create the IAM service object  
var iam = new AWS.IAM({apiVersion: '2010-05-08'});  
  
iam.listAccountAliases({MaxItems: 10}, function(err, data) {  
  if (err) {  
    console.log("Error", err);  
  } else {  
    console.log("Success", data);  
  }  
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListAccountAliases](#) in [AWS SDK for JavaScript API Reference](#).

## List server certificates

The following code example shows how to list IAM server certificates.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";  
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

List the certificates.

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "./libs/iamClient.js";  
import { ListServerCertificatesCommand } from "@aws-sdk/client-iam";
```

```
export const run = async () => {
  try {
    const data = await iamClient.send(new ListServerCertificatesCommand({}));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListServerCertificates](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.listServerCertificates({}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListServerCertificates](#) in [AWS SDK for JavaScript API Reference](#).

## List users

The following code example shows how to list IAM users.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the users.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { ListUsersCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { MaxItems: 10 };

export const run = async () => {
  try {
    const data = await iamClient.send(new ListUsersCommand(params));
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListUsers](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  MaxItems: 10
};

iam.listUsers(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var users = data.Users || [];
    users.forEach(function(user) {
      console.log("User " + user.UserName + " created", user.CreateDate);
    });
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListUsers](#) in [AWS SDK for JavaScript API Reference](#).

## Update a server certificate

The following code example shows how to update an IAM server certificate.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Update a server certificate.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { UpdateServerCertificateCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  ServerCertificateName: "CERTIFICATE_NAME", //CERTIFICATE_NAME
  NewServerCertificateName: "NEW_CERTIFICATE_NAME", //NEW_CERTIFICATE_NAME
};

export const run = async () => {
  try {
    const data = await iamClient.send(
      new UpdateServerCertificateCommand(params)
    );
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateServerCertificate](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
```

```
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  ServerCertificateName: 'CERTIFICATE_NAME',
  NewServerCertificateName: 'NEW_CERTIFICATE_NAME'
};

iam.updateServerCertificate(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateServerCertificate](#) in [AWS SDK for JavaScript API Reference](#).

## Update a user

The following code example shows how to update an IAM user.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Update the user.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { UpdateUserCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  UserName: "ORIGINAL_USER_NAME", //ORIGINAL_USER_NAME
  NewUserName: "NEW_USER_NAME", //NEW_USER_NAME
};

export const run = async () => {
  try {
    const data = await iamClient.send(new UpdateUserCommand(params));
    console.log("Success, username updated");
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateUser](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  UserName: process.argv[2],
  NewUserName: process.argv[3]
};

iam.updateUser(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateUser](#) in [AWS SDK for JavaScript API Reference](#).

## Update an access key

The following code example shows how to update an IAM access key.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Update the access key.

```
// Import required AWS SDK clients and commands for Node.js.
```

```
import { iamClient } from "./libs/iamClient.js";
import { UpdateAccessKeyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  AccessKeyId: "ACCESS_KEY_ID", //ACCESS_KEY_ID
  Status: "Active",
  UserName: "USER_NAME", //USER_NAME
};

export const run = async () => {
  try {
    const data = await iamClient.send(new UpdateAccessKeyCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateAccessKey](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  AccessKeyId: 'ACCESS_KEY_ID',
  Status: 'Active',
  UserName: 'USER_NAME'
};

iam.updateAccessKey(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateAccessKey](#) in [AWS SDK for JavaScript API Reference](#).

## Amazon Pinpoint examples using SDK for JavaScript V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V2 with Amazon Pinpoint.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Topics

- [Actions \(p. 2879\)](#)

## Actions

### Send email and text messages

The following code example shows how to send email and text messages with Amazon Pinpoint.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { PinpointClient } from "@aws-sdk/client-pinpoint";
// Set the AWS Region.
const REGION = "us-east-1";
//Set the MediaConvert Service Object
const pinClient = new PinpointClient({region: REGION});
export { pinClient };
```

Send an email message.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessagesCommand } from "@aws-sdk/client-pinpoint";
import { pinClient } from "./libs/pinClient.js";

// The FromAddress must be verified in SES.
const fromAddress = "FROM_ADDRESS";
const toAddress = "TO_ADDRESS";
const projectId = "PINPOINT_PROJECT_ID";

// The subject line of the email.
var subject = "Amazon Pinpoint Test (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
var body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)

-----
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in Node.js.
For more information, see https://aws.amazon.com/sdk-for-node-js/`;

// The body of the email for recipients whose email clients support HTML content.
var body_html = `<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint Email API</a> using
    the
    <a href='https://aws.amazon.com/sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js
    </a>.</p>

```

```

</body>
</html>';

// The character encoding for the subject line and message body of the email.
var charset = "UTF-8";

const params = {
  ApplicationId: projectId,
  MessageRequest: {
    Addresses: {
      [toAddress]: {
        ChannelType: "EMAIL",
      },
    },
    MessageConfiguration: {
      EmailMessage: {
        FromAddress: fromAddress,
        SimpleEmail: {
          Subject: {
            Charset: charset,
            Data: subject,
          },
          HtmlPart: {
            Charset: charset,
            Data: body_html,
          },
          TextPart: {
            Charset: charset,
            Data: body_text,
          },
        },
      },
    },
  },
};

const run = async () => {
  try {
    const data = await pinClient.send(new SendMessagesCommand(params));

    const {
      MessageResponse: { Result },
    } = data;

    const recipientResult = Result[toAddress];

    if (recipientResult.StatusCode !== 200) {
      throw new Error(recipientResult.StatusMessage);
    } else {
      console.log(recipientResult.MessageId);
    }
  } catch (err) {
    console.log(err.message);
  }
};

run();

```

Send an SMS message.

```

// Import required AWS SDK clients and commands for Node.js
import { SendMessagesCommand } from "@aws-sdk/client-pinpoint";
import { pinClient } from "./libs/pinClient.js";

```

```
("use strict");

/* The phone number or short code to send the message from. The phone number
or short code that you specify has to be associated with your Amazon Pinpoint
account. For best results, specify long codes in E.164 format. */
const originationNumber = "SENDER_NUMBER"; //e.g., +1XXXXXXXXXX

// The recipient's phone number. For best results, you should specify the phone number
// in E.164 format.
const destinationNumber = "RECEIVER_NUMBER"; //e.g., +1XXXXXXXXXX

// The content of the SMS message.
const message =
  "This message was sent through Amazon Pinpoint " +
  "using the AWS SDK for JavaScript in Node.js. Reply STOP to " +
  "opt out./";

/*The Amazon Pinpoint project/application ID to use when you send this message.
Make sure that the SMS channel is enabled for the project or application
that you choose.*/
const projectId = "PINPOINT_PROJECT_ID"; //e.g., XXXXXXXX66e4e9986478cXXXXXXX

/* The type of SMS message that you want to send. If you plan to send
time-sensitive content, specify TRANSACTIONAL. If you plan to send
marketing-related content, specify PROMOTIONAL.*/
var messageType = "TRANSACTIONAL";

// The registered keyword associated with the originating short code.
var registeredKeyword = "myKeyword";

/* The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html.\*

var senderId = "MySenderId";

// Specify the parameters to pass to the API.
var params = {
  ApplicationId: projectId,
  MessageRequest: {
    Addresses: [
      [destinationNumber]: {
        ChannelType: "SMS",
      },
    ],
    MessageConfiguration: {
      SMSMessage: {
        Body: message,
        Keyword: registeredKeyword,
        MessageType: messageType,
        OriginationNumber: originationNumber,
        SenderId: senderId,
      },
    },
  },
};

const run = async () => {
  try {
    const data = await pinClient.send(new SendMessagesCommand(params));
    return data; // For unit tests.
    console.log(
      "Message sent! " +
      data["MessageResponse"]["Result"][destinationNumber]["StatusMessage"]
    );
  }
};
```

```
    } catch (err) {
      console.log(err);
    }
};

run();
```

- For API details, see [SendMessages](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Send an email message.

```
'use strict';

const AWS = require('aws-sdk');

// The AWS Region that you want to use to send the email. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/
const aws_region = "us-west-2"

// The "From" address. This address has to be verified in Amazon Pinpoint
// in the region that you use to send email.
const senderAddress = "sender@example.com";

// The address on the "To" line. If your Amazon Pinpoint account is in
// the sandbox, this address also has to be verified.
var toAddress = "recipient@example.com";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
const appId = "ce796be37f32f178af652b26eexample";

// The subject line of the email.
var subject = "Amazon Pinpoint (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
var body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)
-----
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in Node.js.
For more information, see https://aws.amazon.com/sdk-for-node-js/`;

// The body of the email for recipients whose email clients support HTML content.
var body_html = `<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint API</a> using the
    <a href='https://aws.amazon.com/sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js</a>.</p>
</body>
</html>`;

// The character encoding the you want to use for the subject line and
// message body of the email.
var charset = "UTF-8";
```

```
// Specify that you're using a shared credentials file.  
var credentials = new AWS.SharedIniFileCredentials({profile: 'default'});  
AWS.config.credentials = credentials;  
  
// Specify the region.  
AWS.config.update({region:aws_region});  
  
//Create a new Pinpoint object.  
var pinpoint = new AWS.Pinpoint();  
  
// Specify the parameters to pass to the API.  
var params = {  
    ApplicationId: appId,  
    MessageRequest: {  
        Addresses: {  
            [toAddress]:{  
                ChannelType: 'EMAIL'  
            }  
        },  
        MessageConfiguration: {  
            EmailMessage: {  
                FromAddress: senderAddress,  
                SimpleEmail: {  
                    Subject: {  
                        Charset: charset,  
                        Data: subject  
                    },  
                    HtmlPart: {  
                        Charset: charset,  
                        Data: body_html  
                    },  
                    TextPart: {  
                        Charset: charset,  
                        Data: body_text  
                    }  
                }  
            }  
        }  
    }  
};  
  
//Try to send the email.  
pinpoint.sendMessages(params, function(err, data) {  
    // If something goes wrong, print an error message.  
    if(err) {  
        console.log(err.message);  
    } else {  
        console.log("Email sent! Message ID: ", data['MessageResponse']['Result'][toAddress]['MessageId']);  
    }  
});
```

Send an SMS message.

```
'use strict';  
  
var AWS = require('aws-sdk');  
  
// The AWS Region that you want to use to send the message. For a list of  
// AWS Regions where the Amazon Pinpoint API is available, see  
// https://docs.aws.amazon.com/pinpoint/latest/apireference/.  
var aws_region = "us-east-1";
```

```
// The phone number or short code to send the message from. The phone number
// or short code that you specify has to be associated with your Amazon Pinpoint
// account. For best results, specify long codes in E.164 format.
var originationNumber = "+12065550199";

// The recipient's phone number. For best results, you should specify the
// phone number in E.164 format.
var destinationNumber = "+14255550142";

// The content of the SMS message.
var message = "This message was sent through Amazon Pinpoint "
    + "using the AWS SDK for JavaScript in Node.js. Reply STOP to "
    + "opt out.";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
var applicationId = "ce796be37f32f178af652b26eexample";

// The type of SMS message that you want to send. If you plan to send
// time-sensitive content, specify TRANSACTIONAL. If you plan to send
// marketing-related content, specify PROMOTIONAL.
var messageType = "TRANSACTIONAL";

// The registered keyword associated with the originating short code.
var registeredKeyword = "myKeyword";

// The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
// https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html
var senderId = "MySenderId";

// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({profile: 'default'});
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({region:aws_region});

//Create a new Pinpoint object.
var pinpoint = new AWS.Pinpoint();

// Specify the parameters to pass to the API.
var params = {
    ApplicationId: applicationId,
    MessageRequest: {
        Addresses: [
            [destinationNumber]: {
                ChannelType: 'SMS'
            }
        ],
        MessageConfiguration: {
            SMSMessage: {
                Body: message,
                Keyword: registeredKeyword,
                MessageType: messageType,
                OriginationNumber: originationNumber,
                SenderId: senderId,
            }
        }
    }
};

//Try to send the message.
```

```
pinpoint.sendMessages(params, function(err, data) {
  // If something goes wrong, print an error message.
  if(err) {
    console.log(err.message);
  // Otherwise, show the unique ID for the message.
  } else {
    console.log("Message sent! "
      + data['MessageResponse']['Result'][destinationNumber]['StatusMessage']);
  }
});
```

- For API details, see [SendMessages](#) in *AWS SDK for JavaScript API Reference*.

## Amazon Pinpoint SMS and Voice API examples using SDK for JavaScript V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V2 with Amazon Pinpoint SMS and Voice API.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2885\)](#)

## Actions

### Send a voice message with Amazon Pinpoint SMS and Voice API

The following code example shows how to send a voice message with Amazon Pinpoint SMS and Voice API.

#### SDK for JavaScript V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
'use strict'

var AWS = require('aws-sdk');

// The AWS Region that you want to use to send the voice message. For a list of
// AWS Regions where the Amazon Pinpoint SMS and Voice API is available, see
// https://docs.aws.amazon.com/pinpoint-sms-voice/latest/APIReference/
var aws_region = "us-east-1";

// The phone number that the message is sent from. The phone number that you
// specify has to be associated with your Amazon Pinpoint account. For best results,
// you
// should specify the phone number in E.164 format.
var originationNumber = "+12065550110";
```

```
// The recipient's phone number. For best results, you should specify the phone
// number in E.164 format.
var destinationNumber = "+12065550142";

// The language to use when sending the message. For a list of supported
// languages, see https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
var languageCode = "en-US";

// The Amazon Polly voice that you want to use to send the message. For a list
// of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
var voiceId = "Matthew";

// The content of the message. This example uses SSML to customize and control
// certain aspects of the message, such as the volume or the speech rate.
// The message can't contain any line breaks.
var ssmlMessage = "<speak>
    + \"This is a test message sent from <emphasis>Amazon Pinpoint</emphasis> \""
    + \"using the <break strength='weak' />AWS SDK for JavaScript in Node.js. \""
    + "<amazon:effect phonation='soft'>Thank you for listening."
    + "</amazon:effect>"
    + "</speak>";

// The phone number that you want to appear on the recipient's device. The phone
// number that you specify has to be associated with your Amazon Pinpoint account.
var callerId = "+12065550199";

// The configuration set that you want to use to send the message.
var configurationSet = "ConfigSet";

// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({profile: 'default'});
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({region:aws_region});

//Create a new Pinpoint object.
var pinpointsmsvoice = new AWS.PinpointSMSVoice();

var params = {
  CallerId: callerId,
  ConfigurationSetName: configurationSet,
  Content: {
    SSMLMessage: {
      LanguageCode: languageCode,
      Text: ssmlMessage,
      VoiceId: voiceId
    }
  },
  DestinationPhoneNumber: destinationNumber,
  OriginationPhoneNumber: originationNumber
};

//Try to send the message.
pinpointsmsvoice.sendVoiceMessage(params, function(err, data) {
  // If something goes wrong, print an error message.
  if(err) {
    console.log(err.message);
  // Otherwise, show the unique ID for the message.
  } else {
    console.log("Message sent! Message ID: " + data['MessageId']);
  }
});
```

- For API details, see [SendVoiceMessage](#) in *AWS SDK for JavaScript API Reference*.

## S3 Glacier examples using SDK for JavaScript V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V2 with Amazon S3 Glacier.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2887\)](#)

## Actions

### Create a multipart upload

The following code example shows how to create a multipart upload to an Amazon S3 Glacier vault.

#### SDK for JavaScript V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a multipart upload of 1 megabyte chunks of a Buffer object.

```
// Create a new service object and some supporting variables
var glacier = new AWS.Glacier({apiVersion: '2012-06-01'}),
    vaultName = 'YOUR_VAULT_NAME',
    buffer = new Buffer(2.5 * 1024 * 1024), // 2.5MB buffer
    partSize = 1024 * 1024, // 1MB chunks,
    numPartsLeft = Math.ceil(buffer.length / partSize),
    startTime = new Date(),
    params = {vaultName: vaultName, partSize: partSize.toString()};

// Compute the complete SHA-256 tree hash so we can pass it
// to completeMultipartUpload request at the end
var treeHash = glacier.computeChecksums(buffer).treeHash;

// Initiate the multipart upload
console.log('Initiating upload to', vaultName);
// Call Glacier to initiate the upload.
glacier.initiateMultipartUpload(params, function (mpErr, multipart) {
    if (mpErr) { console.log('Error!', mpErr.stack); return; }
    console.log("Got upload ID", multipart.uploadId);

    // Grab each partSize chunk and upload it as a part
    for (var i = 0; i < buffer.length; i += partSize) {
        var end = Math.min(i + partSize, buffer.length),
            partParams = {
                vaultName: vaultName,
                uploadId: multipart.uploadId,
                range: 'bytes ' + i + '-' + (end-1) + '/',
                body: buffer.slice(i, end)
            };
    }
});
```

```
// Send a single part
console.log('Uploading part', i, '=', partParams.range);
glacier.uploadMultipartPart(partParams, function(multiErr, mData) {
    if (multiErr) return;
    console.log("Completed part", this.request.params.range);
    if (--numPartsLeft > 0) return; // complete only when all parts uploaded

    var doneParams = {
        vaultName: vaultName,
        uploadId: multipart.uploadId,
        archiveSize: buffer.length.toString(),
        checksum: treeHash // the computed tree hash
    };

    console.log("Completing upload...");
    glacier.completeMultipartUpload(doneParams, function(err, data) {
        if (err) {
            console.log("An error occurred while uploading the archive");
            console.log(err);
        } else {
            var delta = (new Date() - startTime) / 1000;
            console.log('Completed upload in', delta, 'seconds');
            console.log('Archive ID:', data.archiveId);
            console.log('Checksum: ', data.checksum);
        }
    });
});
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UploadMultipartPart](#) in [AWS SDK for JavaScript API Reference](#).

## Create a vault

The following code example shows how to create an Amazon S3 Glacier vault.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

Create the vault.

```
// Load the SDK for JavaScript
import { CreateVaultCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
```

```
const vaultname = "VAULT_NAME"; // VAULT_NAME
const params = { vaultName: vaultname };

const run = async () => {
  try {
    const data = await glacierClient.send(new CreateVaultCommand(params));
    console.log("Success, vault created!");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error");
  }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateVault](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the SDK for JavaScript
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create a new service object
var glacier = new AWS.Glacier({apiVersion: '2012-06-01'});
// Call Glacier to create the vault
glacier.createVault({vaultName: 'YOUR_VAULT_NAME'}, function(err) {
  if (!err) {
    console.log("Created vault!")
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateVault](#) in [AWS SDK for JavaScript API Reference](#).

## Upload an archive to a vault

The following code example shows how to upload an archive to an Amazon S3 Glacier vault.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

Upload the archive.

```
// Load the SDK for JavaScript
import { UploadArchiveCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME

// Create a new service object and buffer
const buffer = new Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer
const params = { vaultName: vaultname, body: buffer };

const run = async () => {
  try {
    const data = await glacierClient.send(new UploadArchiveCommand(params));
    console.log("Archive ID", data.archiveId);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error uploading archive!", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UploadArchive](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the SDK for JavaScript
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create a new service object and buffer
var glacier = new AWS.Glacier({apiVersion: '2012-06-01'});
buffer = Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer

var params = {vaultName: 'YOUR_VAULT_NAME', body: buffer};
// Call Glacier to upload the archive.
glacier.uploadArchive(params, function(err, data) {
  if (err) {
    console.log("Error uploading archive!", err);
  } else {
    console.log("Archive ID", data.archiveId);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UploadArchive](#) in [AWS SDK for JavaScript API Reference](#).

## Amazon SNS examples using SDK for JavaScript V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V2 with Amazon Simple Notification Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2891\)](#)

## Actions

### Get the properties of a topic

The following code example shows how to get the properties of an Amazon SNS topic.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {GetTopicAttributesCommand} from "@aws-sdk/client-sns";
import {snsClient} from "./libs/snsClient.js";

// Set the parameters
const params = { TopicArn: "TOPIC_ARN" }; // TOPIC_ARN

const run = async () => {
  try {
    const data = await snsClient.send(new GetTopicAttributesCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetTopicAttributes](#) in [AWS SDK for JavaScript API Reference](#).

#### SDK for JavaScript V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Import the SDK and client modules and call the API.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set region
AWS.config.update({region: 'REGION'});

// Create promise and SNS service object
var getTopicAttrbsPromise = new AWS.SNS({apiVersion:
  '2010-03-31'}).getTopicAttributes({TopicArn: 'TOPIC_ARN'}).promise();

// Handle promise's fulfilled/rejected states
getTopicAttrbsPromise.then(
  function(data) {
    console.log(data);
  }).catch(
  function(err) {
    console.error(err, err.stack);
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetTopicAttributes](#) in [AWS SDK for JavaScript API Reference](#).

## Amazon SQS examples using SDK for JavaScript V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V2 with Amazon Simple Queue Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2892\)](#)

## Actions

### Change how long a queue waits for a message

The following code example shows how to change how long an Amazon SQS queue waits for a message to arrive.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Change how long an Amazon SQS queue waits for a message to arrive.

```
// Import required AWS SDK clients and commands for Node.js
import { SetQueueAttributesCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = {
  Attributes: {
    ReceiveMessageWaitTimeSeconds: "20",
  },
  QueueUrl: "SQS_QUEUE_URL", //SQS_QUEUE_URL; e.g., 'https://sqs.REGION.amazonaws.com/
ACCOUNT-ID/QUEUE-NAME'
};

const run = async () => {
  try {
    const data = await sqsClient.send(new SetQueueAttributesCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.error(err, err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SetQueueAttributes](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Change how long an Amazon SQS queue waits for a message to arrive.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {
  Attributes: {
    "ReceiveMessageWaitTimeSeconds": "20",
  },
  QueueUrl: "SQS_QUEUE_URL"
};

sqs.setQueueAttributes(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [SetQueueAttributes](#) in *AWS SDK for JavaScript API Reference*.

## Change message timeout visibility

The following code example shows how to change an Amazon SQS message timeout visibility.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Receive an Amazon SQS message and change its timeout visibility.

```
// Import required AWS SDK clients and commands for Node.js
import {
  ReceiveMessageCommand,
  ChangeMessageVisibilityCommand,
} from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const queueURL = "https://sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME"; // REGION,
  ACCOUNT_ID, QUEUE_NAME
const params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
};

const run = async () => {
  try {
    const data = await sqsClient.send(new ReceiveMessageCommand(params));
    if (data.Messages != null) {
      try {
        const visibilityParams = {
          QueueUrl: queueURL,
          ReceiptHandle: data.Messages[0].ReceiptHandle,
          VisibilityTimeout: 20, // 20 second timeout
        };
        const results = await sqsClient.send(
          new ChangeMessageVisibilityCommand(visibilityParams)
        );
        console.log("Timeout Changed", results);
      } catch (err) {
        console.log("Delete Error", err);
      }
    } else {
      console.log("No messages to change");
    }
  }
}
```

```
        return data; // For unit tests.
    } catch (err) {
        console.log("Receive Error", err);
    }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ChangeMessageVisibility](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Receive an Amazon SQS message and change its timeout visibility.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk')
// Set the region to us-west-2
AWS.config.update({ region: 'us-west-2' })

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: '2012-11-05' })

var queueURL = 'https://sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'

var params = {
    AttributeNames: ['SentTimestamp'],
    MaxNumberOfMessages: 1,
    MessageAttributeNames: ['All'],
    QueueUrl: queueURL
}

sqs.receiveMessage(params, function (err, data) {
    if (err) {
        console.log('Receive Error', err)
    } else {
        // Make sure we have a message
        if (data.Messages != null) {
            var visibilityParams = {
                QueueUrl: queueURL,
                ReceiptHandle: data.Messages[0].ReceiptHandle,
                VisibilityTimeout: 20 // 20 second timeout
            }
            sqs.changeMessageVisibility(visibilityParams, function (err, data) {
                if (err) {
                    console.log('Delete Error', err)
                } else {
                    console.log('Timeout Changed', data)
                }
            })
        } else {
            console.log('No messages to change')
        }
    }
})
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ChangeMessageVisibility](#) in [AWS SDK for JavaScript API Reference](#).

## Create a queue

The following code example shows how to create an Amazon SQS queue.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Create an Amazon SQS standard queue.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateQueueCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = {
  QueueName: "SQS_QUEUE_NAME", //SQS_QUEUE_URL
  Attributes: {
    DelaySeconds: "60", // Number of seconds delay.
    MessageRetentionPeriod: "86400", // Number of seconds delay.
  },
};

const run = async () => {
  try {
    const data = await sqsClient.send(new CreateQueueCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Create an Amazon SQS queue that waits for a message to arrive.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateQueueCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = {
  QueueName: "SQS_QUEUE_NAME", //SQS_QUEUE_URL; e.g., 'https://
  sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'
  Attributes: {
    ReceiveMessageWaitTimeSeconds: "20",
  },
};
```

```
const run = async () => {
  try {
    const data = await sqsClient.send(new CreateQueueCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.error(err, err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateQueue](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an Amazon SQS standard queue.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {
  QueueName: 'SQS_QUEUE_NAME',
  Attributes: {
    'DelaySeconds': '60',
    'MessageRetentionPeriod': '86400'
  }
};

sqs.createQueue(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

Create an Amazon SQS queue that waits for a message to arrive.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {
  QueueName: 'SQS_QUEUE_NAME',
  Attributes: {
    'ReceiveMessageWaitTimeSeconds': '20',
  }
}
```

```
};

sq.createQueue(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateQueue](#) in *AWS SDK for JavaScript API Reference*.

## Delete a message from a queue

The following code example shows how to delete a message from an Amazon SQS queue.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Receive and delete Amazon SQS messages.

```
// Import required AWS SDK clients and commands for Node.js
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
} from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const queueURL = "SQS_QUEUE_URL"; //SQS_QUEUE_URL; e.g., 'https://
sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'
const params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 10,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  VisibilityTimeout: 20,
  WaitTimeSeconds: 0,
};

const run = async () => {
  try {
    const data = await sqsClient.send(new ReceiveMessageCommand(params));
    if (data.Messages) {
      var deleteParams = {
```

```
        QueueUrl: queueURL,
        ReceiptHandle: data.Messages[0].ReceiptHandle,
    };
    try {
        const data = await sqsClient.send(new DeleteMessageCommand(deleteParams));
        console.log("Message deleted", data);
    } catch (err) {
        console.log("Error", err);
    }
} else {
    console.log("No messages to delete");
}
return data; // For unit tests.
} catch (err) {
    console.log("Receive Error", err);
}
};

run();
};
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteMessage](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Receive and delete Amazon SQS messages.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var queueURL = "SQS_QUEUE_URL";

var params = {
    AttributeNames: [
        "SentTimestamp"
    ],
    MaxNumberOfMessages: 10,
    MessageAttributeNames: [
        "All"
    ],
    QueueUrl: queueURL,
    VisibilityTimeout: 20,
    WaitTimeSeconds: 0
};

sqs.receiveMessage(params, function(err, data) {
    if (err) {
        console.log("Receive Error", err);
    } else if (data.Messages) {
        var deleteParams = {
            QueueUrl: queueURL,
            ReceiptHandle: data.Messages[0].ReceiptHandle
        };
        sqs.deleteMessage(deleteParams, function(err, data) {
            if (err) {
                console.log("Delete Error", err);
            }
        });
    }
});
```

```
        } else {
            console.log("Message Deleted", data);
        }
    });
});
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteMessage](#) in [AWS SDK for JavaScript API Reference](#).

## Delete a queue

The following code example shows how to delete an Amazon SQS queue.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Delete an Amazon SQS queue.

```
// Import required AWS SDK clients and commands for Node.js
import { DeleteQueueCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = { QueueUrl: "SQS_QUEUE_URL" }; //SQS_QUEUE_URL e.g., 'https://
sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'

const run = async () => {
    try {
        const data = await sqsClient.send(new DeleteQueueCommand(params));
        console.log("Success", data);
        return data; // For unit tests.
    } catch (err) {
        console.error(err, err.stack);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteQueue](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an Amazon SQS queue.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {
  QueueUrl: 'SQS_QUEUE_URL'
};

sqs.deleteQueue(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteQueue](#) in [AWS SDK for JavaScript API Reference](#).

### Get the URL of a queue

The following code example shows how to get the URL of an Amazon SQS queue.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Get the URL for an Amazon SQS queue.

```
// Import required AWS SDK clients and commands for Node.js
import { GetQueueUrlCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = { QueueName: "SQS_QUEUE_NAME" };

const run = async () => {
  try {
    const data = await sqsClient.send(new GetQueueUrlCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
```

```
        console.log("Error", err);
    }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetQueueUrl](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get the URL for an Amazon SQS queue.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {
  QueueName: 'SQS_QUEUE_NAME'
};

sqs.getQueueUrl(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetQueueUrl](#) in [AWS SDK for JavaScript API Reference](#).

### List queues

The following code example shows how to list Amazon SQS queues.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

List your Amazon SQS queues.

```
// Import required AWS SDK clients and commands for Node.js
import { ListQueuesCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

const run = async () => {
  try {
    const data = await sqsClient.send(new ListQueuesCommand({}));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.error(err, err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListQueues](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List your Amazon SQS queues.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {};

sqs.listQueues(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrls);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListQueues](#) in [AWS SDK for JavaScript API Reference](#).

### Receive messages from a queue

The following code example shows how to receive messages from an Amazon SQS queue.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Receive a message from an Amazon SQS queue using long-poll support.

```
// Import required AWS SDK clients and commands for Node.js
import { ReceiveMessageCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const queueURL = "SQS_QUEUE_URL"; // SQS_QUEUE_URL
const params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  WaitTimeSeconds: 20,
};

const run = async () => {
  try {
    const data = await sqsClient.send(new ReceiveMessageCommand(params));
    console.log("Success, ", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ReceiveMessage in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Receive a message from an Amazon SQS queue using long-poll support.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var queueURL = "SQS_QUEUE_URL";

var params = {
  AttributeNames: [
    "SentTimestamp"
  ],
  MaxNumberOfMessages: 1,
```

```
    MessageAttributeNames: [
      "All"
    ],
    QueueUrl: queueURL,
    WaitTimeSeconds: 20
  };

  sqs.receiveMessage(params, function(err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  });
}
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ReceiveMessage in AWS SDK for JavaScript API Reference](#).

## Send a message to a queue

The following code example shows how to send a message to an Amazon SQS queue.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Send a message to an Amazon SQS queue.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessageCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = {
  DelaySeconds: 10,
  MessageAttributes: {
    Title: {
      DataType: "String",
      StringValue: "The Whistler",
    },
    Author: {
      DataType: "String",
      StringValue: "John Grisham",
    },
    WeeksOn: {
      DataType: "Number",
      StringValue: "6",
    },
  },
};
```

```
MessageBody:  
    "Information about current NY Times fiction bestseller for week of 12/11/2016.",  
    // MessageDeduplicationId: "TheWhistler", // Required for FIFO queues  
    // MessageGroupId: "Group1", // Required for FIFO queues  
    QueueUrl: "SQS_QUEUE_URL" //SQS_QUEUE_URL; e.g., 'https://sqs.REGION.amazonaws.com/  
ACCOUNT-ID/QUEUE-NAME'  
};  
  
const run = async () => {  
    try {  
        const data = await sqsClient.send(new SendMessageCommand(params));  
        console.log("Success, message sent. MessageID:", data.MessageId);  
        return data; // For unit tests.  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SendMessage](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Send a message to an Amazon SQS queue.

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create an SQS service object  
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});  
  
var params = {  
    // Remove DelaySeconds parameter and value for FIFO queues  
    DelaySeconds: 10,  
    MessageAttributes: {  
        "Title": {  
            DataType: "String",  
            StringValue: "The Whistler"  
        },  
        "Author": {  
            DataType: "String",  
            StringValue: "John Grisham"  
        },  
        "WeeksOn": {  
            DataType: "Number",  
            StringValue: "6"  
        }  
    },  
    MessageBody: "Information about current NY Times fiction bestseller for week of  
12/11/2016.",  
    // MessageDeduplicationId: "TheWhistler", // Required for FIFO queues  
    // MessageGroupId: "Group1", // Required for FIFO queues  
    QueueUrl: "SQS_QUEUE_URL"  
};  
  
sqs.sendMessage(params, function(err, data) {  
    if (err) {
```

```
        console.log("Error", err);
    } else {
        console.log("Success", data.MessageId);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SendMessage](#) in [AWS SDK for JavaScript API Reference](#).

## AWS STS examples using SDK for JavaScript V2

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V2 with AWS Security Token Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2907\)](#)

## Actions

### Assume a role

The following code example shows how to assume a role with AWS STS.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon STS service client object.
const stsClient = new STSClient({ region: REGION });
export { stsClient };
```

Assume the IAM role.

```
// Import required AWS SDK clients and commands for Node.js
import { stsClient } from "./libs/stsClient.js";
import {
    AssumeRoleCommand,
    GetCallerIdentityCommand,
} from "@aws-sdk/client-sts";

// Set the parameters
export const params = {
    RoleArn: "ARN_OF_ROLE_TO_ASSUME", //ARN_OF_ROLE_TO_ASSUME
    RoleSessionName: "session1",
```

```
    DurationSeconds: 900,
};

export const run = async () => {
  try {
    //Assume Role
    const data = await stsClient.send(new AssumeRoleCommand(params));
    return data;
    const rolecreds = {
      accessKeyId: data.Credentials.AccessKeyId,
      secretAccessKey: data.Credentials.SecretAccessKey,
      sessionToken: data.Credentials.SessionToken,
    };
    //Get Amazon Resource Name (ARN) of current identity
    try {
      const stsParams = { credentials: rolecreds };
      const stsClient = new STSClient(stsParams);
      const results = await stsClient.send(
        new GetCallerIdentityCommand(rolecreds)
      );
      console.log("Success", results);
    } catch (err) {
      console.log(err, err.stack);
    }
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [AssumeRole](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
const AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

var roleToAssume = {RoleArn: 'arn:aws:iam::123456789012:role/RoleName',
                    RoleSessionName: 'session1',
                    DurationSeconds: 900,};
var roleCreds;

// Create the STS service object
var sts = new AWS.STS({apiVersion: '2011-06-15'});

//Assume Role
sts.assumeRole(roleToAssume, function(err, data) {
  if (err) console.log(err, err.stack);
  else{
    roleCreds = {accessKeyId: data.Credentials.AccessKeyId,
                secretAccessKey: data.Credentials.SecretAccessKey,
                sessionToken: data.Credentials.SessionToken};
    stsGetCallerIdentity(roleCreds);
  }
});
//Get Arn of current identity
```

```
function stsGetCallerIdentity(creds) {
    var stsParams = {credentials: creds };
    // Create STS service object
    var sts = new AWS.STS(stsParams);

    sts.getCallerIdentity({}, function(err, data) {
        if (err) {
            console.log(err, err.stack);
        }
        else {
            console.log(data.ArN);
        }
    });
}
```

- For API details, see [AssumeRole](#) in *AWS SDK for JavaScript API Reference*.

## Cross-service examples using SDK for JavaScript V2

The following sample applications use the AWS SDK for JavaScript V2 to work across multiple AWS services.

### Examples

- [Invoke a Lambda function from a browser \(p. 2909\)](#)

## Invoke a Lambda function from a browser

### SDK for JavaScript V2

You can create a browser-based application that uses an AWS Lambda function to update an Amazon DynamoDB table with user selections.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Lambda

### SDK for JavaScript V3

You can create a browser-based application that uses an AWS Lambda function to update an Amazon DynamoDB table with user selections. This app uses AWS SDK for JavaScript v3.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Lambda

## Code examples for SDK for JavaScript V3

The code examples in this topic show you how to use the AWS SDK for JavaScript V3 with AWS.

### Examples

- Single-service actions and scenarios using [SDK for JavaScript V3 \(p. 2910\)](#)
- Cross-service examples using [SDK for JavaScript V3 \(p. 3150\)](#)

## Single-service actions and scenarios using SDK for JavaScript V3

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with AWS services.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Services

- [CloudWatch examples using SDK for JavaScript V3 \(p. 2910\)](#)
- [CloudWatch Events examples using SDK for JavaScript V3 \(p. 2922\)](#)
- [CloudWatch Logs examples using SDK for JavaScript V3 \(p. 2926\)](#)
- [Amazon Cognito Identity Provider examples using SDK for JavaScript V3 \(p. 2931\)](#)
- [DynamoDB examples using SDK for JavaScript V3 \(p. 2943\)](#)
- [EventBridge examples using SDK for JavaScript V3 \(p. 2993\)](#)
- [AWS Glue examples using SDK for JavaScript V3 \(p. 2997\)](#)
- [IAM examples using SDK for JavaScript V3 \(p. 3012\)](#)
- [Lambda examples using SDK for JavaScript V3 \(p. 3055\)](#)
- [Amazon Personalize examples using SDK for JavaScript V3 \(p. 3061\)](#)
- [Amazon Personalize Events examples using SDK for JavaScript V3 \(p. 3072\)](#)
- [Amazon Personalize Runtime examples using SDK for JavaScript V3 \(p. 3074\)](#)
- [Amazon Pinpoint examples using SDK for JavaScript V3 \(p. 3077\)](#)
- [Amazon Redshift examples using SDK for JavaScript V3 \(p. 3083\)](#)
- [Amazon S3 examples using SDK for JavaScript V3 \(p. 3087\)](#)
- [S3 Glacier examples using SDK for JavaScript V3 \(p. 3111\)](#)
- [Amazon SES examples using SDK for JavaScript V3 \(p. 3114\)](#)
- [Amazon SNS examples using SDK for JavaScript V3 \(p. 3115\)](#)
- [Amazon SQS examples using SDK for JavaScript V3 \(p. 3127\)](#)
- [AWS STS examples using SDK for JavaScript V3 \(p. 3142\)](#)
- [Amazon Transcribe examples using SDK for JavaScript V3 \(p. 3144\)](#)

## CloudWatch examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon CloudWatch.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2911\)](#)

## Actions

### Create an alarm that watches a metric

The following code example shows how to create an Amazon CloudWatch alarm that watches a metric.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { PutMetricAlarmCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanOrEqualToThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: false,
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

export const run = async () => {
  try {
    const data = await cwClient.send(new PutMetricAlarmCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutMetricAlarm in AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
    AlarmName: 'Web_Server_CPU_Utilization',
    ComparisonOperator: 'GreaterThanOrEqualToThreshold',
    EvaluationPeriods: 1,
    MetricName: 'CPUUtilization',
    Namespace: 'AWS/EC2',
    Period: 60,
    Statistic: 'Average',
    Threshold: 70.0,
    ActionsEnabled: false,
    AlarmDescription: 'Alarm when server CPU exceeds 70%',
    Dimensions: [
        {
            Name: 'InstanceId',
            Value: 'INSTANCE_ID'
        },
    ],
    Unit: 'Percent'
};

cw.putMetricAlarm(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutMetricAlarm in AWS SDK for JavaScript API Reference](#).

### Delete alarms

The following code example shows how to delete Amazon CloudWatch alarms.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { DeleteAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = { AlarmNames: "ALARM_NAME" }; // e.g.,
  "Web_Server_CPU_Utilization"

export const run = async () => {
  try {
    const data = await cwClient.send(new DeleteAlarmsCommand(params));
    console.log("Success, alarm deleted; requestID:", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};

// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAlarms](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Import the SDK and client modules and call the API.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
  AlarmNames: ['Web_Server_CPU_Utilization']
};

cw.deleteAlarms(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAlarms](#) in [AWS SDK for JavaScript API Reference](#).

## Describe alarms for a metric

The following code example shows how to describe Amazon CloudWatch alarms for a metric.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { DescribeAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = { StateValue: "INSUFFICIENT_DATA" };

export const run = async () => {
  try {
    const data = await cwClient.send(new DescribeAlarmsCommand(params));
    console.log("Success", data);
    data.MetricAlarms.forEach(function (item) {
      console.log(item.AlarmName);
    });
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeAlarmsForMetric](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

cw.describeAlarms({StateValue: 'INSUFFICIENT_DATA'}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    // List the names of all current alarms in the console
    data.MetricAlarms.forEach(function (item, index, array) {
      console.log(item.AlarmName);
    });
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeAlarmsForMetric](#) in *AWS SDK for JavaScript API Reference*.

## Disable alarm actions

The following code example shows how to disable Amazon CloudWatch alarm actions.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { DisableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = { AlarmNames: "ALARM_NAME" }; // e.g.,
// "Web_Server_CPU_Utilization"

export const run = async () => {
  try {
    const data = await cwClient.send(new DisableAlarmActionsCommand(params));
    console.log("Success, alarm disabled:", data);
    return data;
  } catch (err) {
```

```
        console.log("Error", err);
    }
};

// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DisableAlarmActions](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Import the SDK and client modules and call the API.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

cw.disableAlarmActions({AlarmNames: ['Web_Server_CPU_Utilization']}, function(err,
  data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DisableAlarmActions](#) in *AWS SDK for JavaScript API Reference*.

## Enable alarm actions

The following code example shows how to enable Amazon CloudWatch alarm actions.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {
  PutMetricAlarmCommand,
  EnableAlarmActionsCommand,
} from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = {
  AlarmName: "ALARM_NAME", //ALARM_NAME
  ComparisonOperator: "GreaterThanOrEqualToThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: true,
  AlarmActions: ["ACTION_ARN"], //e.g., "arn:aws:automate:us-east-1:ec2:stop"
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

export const run = async () => {
  try {
    const data = await cwClient.send(new PutMetricAlarmCommand(params));
    console.log("Alarm action added; RequestID:", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
}

try {
  const data = await cwClient.send(
    new EnableAlarmActionsCommand(paramsEnableAlarmAction)
  );
  console.log("Alarm action enabled; RequestID:", data.$metadata.requestId);
} catch (err) {
  console.log("Error", err);
}

// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [EnableAlarmActions](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Import the SDK and client modules and call the API.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
    AlarmName: 'Web_Server_CPU_Utilization',
    ComparisonOperator: 'GreaterThanOrEqualToThreshold',
    EvaluationPeriods: 1,
    MetricName: 'CPUUtilization',
    Namespace: 'AWS/EC2',
    Period: 60,
    Statistic: 'Average',
    Threshold: 70.0,
    ActionsEnabled: true,
    AlarmActions: ['ACTION_ARN'],
    AlarmDescription: 'Alarm when server CPU exceeds 70%',
    Dimensions: [
        {
            Name: 'InstanceId',
            Value: 'INSTANCE_ID'
        },
    ],
    Unit: 'Percent'
};

cw.putMetricAlarm(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Alarm action added", data);
        var paramsEnableAlarmAction = {
            AlarmNames: [params.AlarmName]
        };
        cw.enableAlarmActions(paramsEnableAlarmAction, function(err, data) {
            if (err) {
                console.log("Error", err);
            } else {
                console.log("Alarm action enabled", data);
            }
        });
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [EnableAlarmActions](#) in [AWS SDK for JavaScript API Reference](#).

## List metrics

The following code example shows how to list Amazon CloudWatch metrics.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { ListMetricsCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = {
  Dimensions: [
    {
      Name: "LogGroupName" /* required */,
    },
  ],
  MetricName: "IncomingLogEvents",
  Namespace: "AWS/Logs",
};

export const run = async () => {
  try {
    const data = await cwClient.send(new ListMetricsCommand(params));
    console.log("Success. Metrics:", JSON.stringify(data.Metrics));
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListMetrics](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

var params = {
  Dimensions: [
    {
      Name: 'LogGroupName', /* required */
    },
  ],
  MetricName: "IncomingLogEvents",
  Namespace: "AWS/Logs",
};
```

```
        },
    ],
    MetricName: 'IncomingLogEvents',
    Namespace: 'AWS/Logs'
};

cw.listMetrics(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Metrics", JSON.stringify(data.Metrics));
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListMetrics](#) in [AWS SDK for JavaScript API Reference](#).

### Put data into a metric

The following code example shows how to put data into a Amazon CloudWatch metric.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cwClient = new CloudWatchClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { PutMetricDataCommand } from "@aws-sdk/client-cloudwatch";
import { cwClient } from "./libs/cloudWatchClient.js";

// Set the parameters
export const params = {
    MetricData: [
        {
            MetricName: "PAGES_VISITED",
            Dimensions: [
                {
                    Name: "UNIQUE_PAGES",
                    Value: "URLS",
                },
            ],
            Unit: "None",
            Value: 1.0,
        },
    ],
},
```

```
    Namespace: "SITE/TRAFFIC",
};

export const run = async () => {
  try {
    const data = await cwClient.send(new PutMetricDataCommand(params));
    console.log("Success", data.$metadata.requestId);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutMetricData](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatch service object
var cw = new AWS.CloudWatch({apiVersion: '2010-08-01'});

// Create parameters JSON for putMetricData
var params = {
  MetricData: [
    {
      MetricName: 'PAGES_VISITED',
      Dimensions: [
        {
          Name: 'UNIQUE_PAGES',
          Value: 'URLS'
        },
      ],
      Unit: 'None',
      Value: 1.0
    },
  ],
  Namespace: 'SITE/TRAFFIC'
};

cw.putMetricData(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", JSON.stringify(data));
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutMetricData](#) in [AWS SDK for JavaScript API Reference](#).

## CloudWatch Events examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon CloudWatch Events.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2922\)](#)

## Actions

### Adding a Lambda function target

The following code example shows how to add an AWS Lambda function target to an Amazon CloudWatch Events event.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cweClient = new CloudWatchEventsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { PutTargetsCommand } from "@aws-sdk/client-cloudwatch-events";
import { cweClient } from "./libs/cloudWatchEventsClient.js";

// Set the parameters
export const params = {
  Rule: "DEMO_EVENT",
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN", //LAMBDA_FUNCTION_ARN
      Id: "myCloudWatchEventsTarget",
    },
  ],
};

export const run = async () => {
  try {
    const data = await cweClient.send(new PutTargetsCommand(params));
    console.log("Success, target added; requestID: ", data);
    return data; // For unit tests.
  } catch (err) {
```

```
        console.log("Error", err);
    }
};

// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutTargets](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({apiVersion: '2015-10-07'});

var params = {
    Rule: 'DEMO_EVENT',
    Targets: [
        {
            Arn: 'LAMBDA_FUNCTION_ARN',
            Id: 'myCloudWatchEventsTarget'
        }
    ]
};

cwevents.putTargets(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutTargets](#) in [AWS SDK for JavaScript API Reference](#).

## Create a scheduled rule

The following code example shows how to create an Amazon CloudWatch Events scheduled rule.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";
// Set the AWS Region.
```

```
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cweClient = new CloudWatchEventsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { PutRuleCommand } from "@aws-sdk/client-cloudwatch-events";
import { cweClient } from "./libs/cloudWatchEventsClient.js";

// Set the parameters
export const params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN", //IAM_ROLE_ARN
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

export const run = async () => {
  try {
    const data = await cweClient.send(new PutRuleCommand(params));
    console.log("Success, scheduled rule created; Rule ARN:", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutRule in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({apiVersion: '2015-10-07'});

var params = {
  Name: 'DEMO_EVENT',
  RoleArn: 'IAM_ROLE_ARN',
  ScheduleExpression: 'rate(5 minutes)',
  State: 'ENABLED'
};

cwevents.putRule(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.RuleArn);
  }
})
```

```
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutRule](#) in [AWS SDK for JavaScript API Reference](#).

## Send events

The following code example shows how to send Amazon CloudWatch Events events.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchEventsClient } from "@aws-sdk/client-cloudwatch-events";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch service client object.
export const cweClient = new CloudWatchEventsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { PutEventsCommand } from "@aws-sdk/client-cloudwatch-events";
import { cweClient } from "./libs/cloudWatchEventsClient.js";

// Set the parameters
export const params = {
  Entries: [
    {
      Detail: '{ "key1": "value1", "key2": "value2" }',
      DetailType: "appRequestSubmitted",
      Resources: [
        "RESOURCE_ARN", //RESOURCE_ARN
      ],
      Source: "com.company.app",
    },
  ],
};

export const run = async () => {
  try {
    const data = await cweClient.send(new PutEventsCommand(params));
    console.log("Success, event sent; requestID:", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [PutEvents](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var cwevents = new AWS.CloudWatchEvents({apiVersion: '2015-10-07'});

var params = {
  Entries: [
    {
      Detail: '{ \"key1\": \"value1\", \"key2\": \"value2\" }',
      DetailType: 'appRequestSubmitted',
      Resources: [
        'RESOURCE_ARN',
      ],
      Source: 'com.company.app'
    }
  ]
};

cwevents.putEvents(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutEvents](#) in *AWS SDK for JavaScript API Reference*.

## CloudWatch Logs examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon CloudWatch Logs.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2926\)](#)

## Actions

### Create a subscription filter

The following code example shows how to create an Amazon CloudWatch Logs subscription filter.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch Logs service client object.
export const cwlClient = new CloudWatchLogsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {
  PutSubscriptionFilterCommand,
} from "@aws-sdk/client-cloudwatch-logs";
import { cwlClient } from "./libs/cloudWatchLogsClient.js";

// Set the parameters
export const params = {
  destinationArn: "LAMBDA_FUNCTION_ARN", //LAMBDA_FUNCTION_ARN
  filterName: "FILTER_NAME", //FILTER_NAME
  filterPattern: "ERROR",
  logGroupName: "LOG_GROUP", //LOG_GROUP
};

export const run = async () => {
  try {
    const data = await cwlClient.send(new PutSubscriptionFilterCommand(params));
    console.log("Success", data.subscriptionFilters);
    return data; //For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutSubscriptionFilter](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({apiVersion: '2014-03-28'});
```

```
var params = {
  destinationArn: 'LAMBDA_FUNCTION_ARN',
  filterName: 'FILTER_NAME',
  filterPattern: 'ERROR',
  logGroupName: 'LOG_GROUP',
};

cwl.putSubscriptionFilter(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutSubscriptionFilter](#) in [AWS SDK for JavaScript API Reference](#).

## Delete a subscription filter

The following code example shows how to delete an Amazon CloudWatch Logs subscription filter.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch Logs service client object.
export const cwlClient = new CloudWatchLogsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {
  DeleteSubscriptionFilterCommand
} from "@aws-sdk/client-cloudwatch-logs";
import { cwlClient } from "./libs/cloudWatchLogsClient.js";

// Set the parameters
export const params = {
  filterName: "FILTER", //FILTER
  logGroupName: "LOG_GROUP", //LOG_GROUP
};

export const run = async () => {
  try {
    const data = await cwlClient.send(
      new DeleteSubscriptionFilterCommand(params)
    );
    console.log(
      "Success, subscription filter deleted",
    );
  } catch (err) {
    console.error(`Error: ${err}`);
  }
};
```

```
        data
    );
    return data; //For unit tests.
} catch (err) {
    console.log("Error", err);
}
};

// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteSubscriptionFilter](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({apiVersion: '2014-03-28'});

var params = {
    filterName: 'FILTER',
    logGroupName: 'LOG_GROUP'
};

cwl.deleteSubscriptionFilter(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteSubscriptionFilter](#) in *AWS SDK for JavaScript API Reference*.

## Describe existing subscription filters

The following code example shows how to describe Amazon CloudWatch Logs existing subscription filters.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { CloudWatchLogsClient } from "@aws-sdk/client-cloudwatch-logs";
// Set the AWS Region.
```

```
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon CloudWatch Logs service client object.
export const cwlClient = new CloudWatchLogsClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import { DescribeSubscriptionFiltersCommand } from "@aws-sdk/client-cloudwatch-logs";
import { cwlClient } from "./libs/cloudWatchLogsClient.js";

// Set the parameters
export const params = {
  logGroupName: "GROUP_NAME", //GROUP_NAME
  limit: 5
};

export const run = async () => {
  try {
    const data = await cwlClient.send(
      new DescribeSubscriptionFiltersCommand(params)
    );
    console.log("Success", data.subscriptionFilters);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeSubscriptionFilters](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the CloudWatchLogs service object
var cwl = new AWS.CloudWatchLogs({apiVersion: '2014-03-28'});

var params = {
  logGroupName: 'GROUP_NAME',
  limit: 5
};

cwl.describeSubscriptionFilters(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.subscriptionFilters);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeSubscriptionFilters](#) in [AWS SDK for JavaScript API Reference](#).

## Amazon Cognito Identity Provider examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon Cognito Identity Provider.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2931\)](#)
- [Scenarios \(p. 2937\)](#)

## Actions

### Confirm a user

The following code example shows how to confirm an Amazon Cognito user.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const confirmSignUp = async ({ clientId, username, code }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};
```

- For API details, see [ConfirmSignUp](#) in [AWS SDK for JavaScript API Reference](#).

### Confirm an MFA device for tracking

The following code example shows how to confirm an MFA device for tracking by Amazon Cognito.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const confirmDevice = async ({ deviceKey, accessToken, passwordVerifier, salt }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new ConfirmDeviceCommand({
    DeviceKey: deviceKey,
    AccessToken: accessToken,
    DeviceSecretVerifierConfig: {
      PasswordVerifier: passwordVerifier,
      Salt: salt
    },
  });

  return client.send(command);
};
```

- For API details, see [ConfirmDevice](#) in *AWS SDK for JavaScript API Reference*.

## Get a token to associate an MFA application with a user

The following code example shows how to get a token to associate an MFA application with an Amazon Cognito user.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const associateSoftwareToken = async (session) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);
  const command = new AssociateSoftwareTokenCommand({
    Session: session,
  });

  return client.send(command);
};
```

- For API details, see [AssociateSoftwareToken](#) in *AWS SDK for JavaScript API Reference*.

## Get information about a user

The following code example shows how to get information about an Amazon Cognito user.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const admin GetUser = async ({ userPoolId, username }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new AdminGetUserCommand({
    UserPoolId: userPoolId,
    Username: username,
  });
};
```

```
    return client.send(command);
};
```

- For API details, see [Admin GetUser](#) in *AWS SDK for JavaScript API Reference*.

## List users

The following code example shows how to list Amazon Cognito users.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const listUsers = async ({ userPoolId }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new ListUsersCommand({
    UserPoolId: userPoolId,
  });

  return client.send(command);
};
```

- For API details, see [ListUsers](#) in *AWS SDK for JavaScript API Reference*.

## Resend a confirmation code

The following code example shows how to resend an Amazon Cognito confirmation code.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const resendConfirmationCode = async ({ clientId, username }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new ResendConfirmationCodeCommand({
    ClientId: clientId,
    Username: username
  });

  return client.send(command);
};
```

- For API details, see [ResendConfirmationCode](#) in *AWS SDK for JavaScript API Reference*.

## Respond to SRP authentication challenges

The following code example shows how to respond to Amazon Cognito SRP authentication challenges.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const respondToAuthChallenge = async ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- For API details, see [RespondToAuthChallenge](#) in *AWS SDK for JavaScript API Reference*.

## Respond to an authentication challenge

The following code example shows how to respond to an Amazon Cognito authentication challenge.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const adminRespondToAuthChallenge = async ({
  userPoolId,
  clientId,
  username,
  totp,
  session,
}) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);
  const command = new AdminRespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: totp,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });
};
```

```
    return client.send(command);
};
```

- For API details, see [AdminRespondToAuthChallenge](#) in *AWS SDK for JavaScript API Reference*.

## Sign up a user

The following code example shows how to sign up a user with Amazon Cognito.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const signUp = async ({ clientId, username, password, email }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};
```

- For API details, see [SignUp](#) in *AWS SDK for JavaScript API Reference*.

## Start authentication with a tracked device

The following code example shows how to start authentication with a device tracked by Amazon Cognito.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const initiateAuth = async ({ username, password, clientId }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new InitiateAuthCommand({
    AuthFlow: AuthFlowType.USER_PASSWORD_AUTH,
    AuthParameters: {
      USERNAME: username,
      PASSWORD: password,
    },
    ClientId: clientId,
  });

  return client.send(command);
};
```

- For API details, see [InitiateAuth](#) in *AWS SDK for JavaScript API Reference*.

### Start authentication with administrator credentials

The following code example shows how to start authentication with Amazon Cognito and administrator credentials.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const adminInitiateAuth = async ({  
    clientId,  
    userPoolId,  
    username,  
    password,  
) => {  
    const client = createClientForDefaultRegion(CognitoIdentityProviderClient);  
  
    const command = new AdminInitiateAuthCommand({  
        ClientId: clientId,  
        UserPoolId: userPoolId,  
        AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,  
        AuthParameters: { USERNAME: username, PASSWORD: password },  
    });  
  
    return client.send(command);  
};
```

- For API details, see [AdminInitiateAuth](#) in *AWS SDK for JavaScript API Reference*.

### Verify an MFA application with a user

The following code example shows how to verify an MFA application with an Amazon Cognito user.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const verifySoftwareToken = async (totp) => {  
    const client = createClientForDefaultRegion(CognitoIdentityProviderClient);  
    // The 'Session' is provided in the response to 'AssociateSoftwareToken'.  
    const session = process.env.SESSION;  
  
    if (!session) {  
        throw new Error(  
            "Missing a valid Session. Did you run 'admin-initiate-auth'?"  
        );  
    }  
  
    const command = new VerifySoftwareTokenCommand({  
        Session: session,  
        UserCode: totp,  
    });
```

```
    return client.send(command);
};
```

- For API details, see [VerifySoftwareToken](#) in *AWS SDK for JavaScript API Reference*.

## Scenarios

### Sign up a user with a user pool that requires MFA

The following code example shows how to:

- Sign up a user with a user name, password, and email address.
- Confirm the user from a code sent in email.
- Set up multi-factor authentication by associating an MFA application with the user.
- Sign in by using a password and an MFA code.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

For the best experience, clone the GitHub repository and run this example. The following code represents a sample of the full example application.

```
import { log } from "../../libs/utils/util-log.js";
import { signUp } from "../../actions/sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "../../libs/utils/util-csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`
    );
  }
};

const validateUser = (username, password, email) => {
  if (!(username && password && email)) {
    throw new Error(
      `Username, password, and email must be provided as arguments to the 'sign-up' command.`
    );
  }
};

const signUpHandler = async (commands) => {
  const [_, username, password, email] = commands;

  try {
    validateUser(username, password, email);
    const clientId = getSecondValuesFromEntries(FILE_USER_POOLS)[0];
    validateClient(clientId);
    log(`Signing up.`);
    await signUp({ clientId, username, password, email });
    log(`Signed up. An confirmation email has been sent to: ${email}.`);
    log(`Run 'confirm-sign-up ${username} <code>' to confirm your account.`);
  } catch (err) {
    log(err);
  }
};
```

```
    }

};

export { signUpHandler };

const signUp = async ({ clientId, username, password, email }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};

import { log } from "../../../../../libs/utils/util-log.js";
import { confirmSignUp } from "../../../../../actions/confirm-sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "../../../../../libs/utils/util-csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`
    );
  }
};

const validateUser = (username) => {
  if (!username) {
    throw new Error(
      `Username name is missing. It must be provided as an argument to the 'confirm-sign-up' command.`
    );
  }
};

const validateCode = (code) => {
  if (!code) {
    throw new Error(
      `Verification code is missing. It must be provided as an argument to the 'confirm-sign-up' command.`
    );
  }
};

const confirmSignUpHandler = async (commands) => {
  const [_, username, code] = commands;

  try {
    validateUser(username);
    validateCode(code);
    const clientId = getSecondValuesFromEntries(FILE_USER_POOLS)[0];
    validateClient(clientId);
    log(`Confirming user.`);
    await confirmSignUp({ clientId, username, code });
    log(
      `User confirmed. Run 'admin-initiate-auth ${username} <password>' to sign in.`
    );
  } catch (err) {
    log(err);
  }
};


```

```
export { confirmSignUpHandler };

const confirmSignUp = async ({ clientId, username, code }) => {
    const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

    const command = new ConfirmSignUpCommand({
        ClientId: clientId,
        Username: username,
        ConfirmationCode: code,
    });

    return client.send(command);
};

import qrcode from "qrcode-terminal";
import { log } from "../../../../../libs/utils/util-log.js";
import { adminInitiateAuth } from "../../../../../actions/admin-initiate-auth.js";
import { associateSoftwareToken } from "../../../../../actions/associate-software-token.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getFirstEntry } from "../../../../../libs/utils/util-csv.js";

const handleMfaSetup = async (session, username) => {
    const { SecretCode, Session } = await associateSoftwareToken(session);

    // Store the Session for use with 'VerifySoftwareToken'.
    process.env.SESSION = Session;

    console.log(
        "Scan this code in your preferred authenticator app, then run 'verify-software-
token' to finish the setup."
    );
    qrcode.generate(
        `otpauth://totp/${username}?secret=${SecretCode}`,
        { small: true },
        console.log
    );
};

const handleSoftwareTokenMfa = (session) => {
    // Store the Session for use with 'AdminRespondToAuthChallenge'.
    process.env.SESSION = session;
};

const validateClient = (id) => {
    if (!id) {
        throw new Error(
            `User pool client id is missing. Did you run 'create-user-pool?'`
        );
    }
};

const validateId = (id) => {
    if (!id) {
        throw new Error(`User pool id is missing. Did you run 'create-user-pool?'`);
    }
};

const validateUser = (username, password) => {
    if (!(username && password)) {
        throw new Error(
            `Username and password must be provided as arguments to the 'admin-initiate-auth'-
command.`
        );
    }
};
```

```
const adminInitiateAuthHandler = async (commands) => {
  const [_, username, password] = commands;

  try {
    validateUser(username, password);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    validateId(userPoolId);
    validateClient(clientId);

    log("Signing in.");
    const { ChallengeName, Session } = await adminInitiateAuth({
      clientId,
      userPoolId,
      username,
      password,
    });

    if (ChallengeName === "MFA_SETUP") {
      log("MFA setup is required.");
      return handleMfaSetup(Session, username);
    }

    if (ChallengeName === "SOFTWARE_TOKEN_MFA") {
      handleSoftwareTokenMfa(Session);
      log(`Run 'admin-respond-to-auth-challenge ${username} <totp>'`);
    }
  } catch (err) {
    log(err);
  }
};

export { adminInitiateAuthHandler };

const adminInitiateAuth = async ({
  clientId,
  userPoolId,
  username,
  password,
}) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};

import { log } from "../../libs/utils/util-log.js";
import { adminRespondToAuthChallenge } from "../../actions/admin-respond-to-auth-challenge.js";
import { getFirstEntry } from "../../libs/utils/util-csv.js";
import { FILE_USER_POOLS } from "./constants.js";

const verifyUsername = (username) => {
  if (!username) {
    throw new Error(
      `Username is missing. It must be provided as an argument to the 'admin-respond-to-auth-challenge' command.`
    );
  }
}
```

```
};

const verifyTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) is missing. It must be provided as an argument to the 'admin-respond-to-auth-challenge' command.`
    );
  }
};

const storeAccessToken = (token) => {
  process.env.AccessToken = token;
};

const adminRespondToAuthChallengeHandler = async (commands) => {
  const [_, username, totp] = commands;

  try {
    verifyUsername(username);
    verifyTotp(totp);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    const session = process.env.SESSION;

    const { AuthenticationResult } = await adminRespondToAuthChallenge({
      clientId,
      userPoolId,
      username,
      totp,
      session,
    });

    storeAccessToken(AuthenticationResult.AccessToken);

    log("Successfully authenticated.");
  } catch (err) {
    log(err);
  }
};

export { adminRespondToAuthChallengeHandler };

const respondToAuthChallenge = async ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

```
import { log } from "../../../../../libs/utils/util-log.js";
import { verifySoftwareToken } from "../../../../../actions/verify-software-token.js";

const validateTotp = (totp) => {
    if (!totp) {
        throw new Error(
            `Time-based one-time password (TOTP) must be provided to the 'validate-software-token' command.
        );
    }
};

const verifySoftwareTokenHandler = async (commands) => {
    const [_, totp] = commands;

    try {
        validateTotp(totp);

        log("Verifying TOTP.");
        await verifySoftwareToken(totp);
        log("TOTP Verified. Run 'admin-initiate-auth' again to sign-in.");
    } catch (err) {
        console.log(err);
    }
};

export { verifySoftwareTokenHandler };

const verifySoftwareToken = async (totp) => {
    const client = createClientForDefaultRegion(CognitoIdentityProviderClient);
    // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
    const session = process.env.SESSION;

    if (!session) {
        throw new Error(
            "Missing a valid Session. Did you run 'admin-initiate-auth'?"
        );
    }

    const command = new VerifySoftwareTokenCommand({
        Session: session,
        UserCode: totp,
    });

    return client.send(command);
};
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.
  - [Admin GetUser](#)
  - [Admin Initiate Auth](#)
  - [Admin Respond To Auth Challenge](#)
  - [Associate Software Token](#)
  - [Confirm Device](#)
  - [Confirm Sign Up](#)
  - [Initiate Auth](#)
  - [List Users](#)
  - [Resend Confirmation Code](#)
  - [Respond To Auth Challenge](#)
  - [Sign Up](#)
  - [Verify Software Token](#)

## DynamoDB examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon DynamoDB.

**Actions** are code excerpts that show you how to call individual service functions.

**Scenarios** are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2943\)](#)
- [Scenarios \(p. 2977\)](#)

## Actions

### Create a table

The following code example shows how to create a DynamoDB table.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create an Amazon DynamoDB service client object.  
const ddbClient = new DynamoDBClient({ region: REGION });  
export { ddbClient };
```

Create the table.

```
// Import required AWS SDK clients and commands for Node.js  
import { CreateTableCommand } from "@aws-sdk/client-dynamodb";  
import { ddbClient } from "./libs/ddbClient.js";  
  
// Set the parameters  
export const params = {  
    AttributeDefinitions: [  
        {  
            AttributeName: "Season", //ATTRIBUTE_NAME_1  
            AttributeType: "N", //ATTRIBUTE_TYPE  
        },  
        {  
            AttributeName: "Episode", //ATTRIBUTE_NAME_2  
            AttributeType: "N", //ATTRIBUTE_TYPE  
        },  
    ],  
    KeySchema: [  
        {  
            AttributeName: "Season", //ATTRIBUTE_NAME_1  
            KeyType: "HASH",
```

```
        },
        {
            AttributeName: "Episode", //ATTRIBUTE_NAME_2
            KeyType: "RANGE",
        },
    ],
    ProvisionedThroughput: {
        ReadCapacityUnits: 1,
        WriteCapacityUnits: 1,
    },
    TableName: "TEST_TABLE", //TABLE_NAME
    StreamSpecification: {
        StreamEnabled: false,
    },
};

export const run = async () => {
    try {
        const data = await ddbClient.send(new CreateTableCommand(params));
        console.log("Table Created", data);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateTable](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
    AttributeDefinitions: [
        {
            AttributeName: 'CUSTOMER_ID',
            AttributeType: 'N'
        },
        {
            AttributeName: 'CUSTOMER_NAME',
            AttributeType: 'S'
        }
    ],
    KeySchema: [
        {
            AttributeName: 'CUSTOMER_ID',
            KeyType: 'HASH'
        },
        {
            AttributeName: 'CUSTOMER_NAME',
            KeyType: 'RANGE'
        }
    ]
};
```

```
        },
    ],
    ProvisionedThroughput: {
        ReadCapacityUnits: 1,
        WriteCapacityUnits: 1
    },
    TableName: 'CUSTOMER_LIST',
    StreamSpecification: {
        StreamEnabled: false
    }
};

// Call DynamoDB to create the table
ddb.createTable(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Table Created", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateTable](#) in [AWS SDK for JavaScript API Reference](#).

## Delete a table

The following code example shows how to delete a DynamoDB table.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../libs/utils/util-aws-sdk.js";
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Delete the table.

```
// Import required AWS SDK clients and commands for Node.js
import { DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import { ddbClient } from "./libs/ddbClient.js";

// Set the parameters
export const params = {
    TableName: "CUSTOMER_LIST_NEW",
};

export const run = async () => {
    try {
        const data = await ddbClient.send(new DeleteTableCommand(params));
        console.log("Success, table deleted", data);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
```

```
    }
};

run();
```

- For API details, see [DeleteTable](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
  TableName: process.argv[2]
};

// Call DynamoDB to delete the specified table
ddb.deleteTable(params, function(err, data) {
  if (err && err.code === 'ResourceNotFoundException') {
    console.log("Error: Table not found");
  } else if (err && err.code === 'ResourceInUseException') {
    console.log("Error: Table in use");
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteTable](#) in *AWS SDK for JavaScript API Reference*.

## Delete an item from a table

The following code example shows how to delete an item from a DynamoDB table.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../../../libs/utils/util-aws-sdk.js";
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the document client.

```
// Create a service client module using ES6 syntax.  
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";  
import { ddbClient } from "./ddbClient.js";  
  
const marshallOptions = {  
    // Whether to automatically convert empty strings, blobs, and sets to `null`.  
    convertEmptyValues: false, // false, by default.  
    // Whether to remove undefined values while marshalling.  
    removeUndefinedValues: true, // false, by default.  
    // Whether to convert typeof object to map attribute.  
    convertClassInstanceToMap: false, // false, by default.  
};  
  
const unmarshallOptions = {  
    // Whether to return numbers as a string instead of converting them to native  
    // JavaScript numbers.  
    wrapNumbers: false, // false, by default.  
};  
  
// Create the DynamoDB document client.  
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {  
    marshallOptions,  
    unmarshallOptions,  
});  
  
export { ddbDocClient };
```

Delete an item from a table using the DynamoDB document client.

```
import { DeleteCommand } from "@aws-sdk/lib-dynamodb";  
import { ddbDocClient } from "../libs/ddbDocClient.js";  
  
// Set the parameters.  
export const params = {  
    TableName: "TABLE_NAME",  
    Key: {  
        primaryKey: "VALUE_1",  
        sortKey: "VALUE_2",  
    },  
};  
  
export const deleteItem = async () => {  
    try {  
        await ddbDocClient.send(new DeleteCommand(params));  
        console.log("Success - item deleted");  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
deleteItem();
```

Delete an item from a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.  
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";  
import { ddbDocClient } from "../libs/ddbDocClient.js";  
  
const tableName = process.argv[2];
```

```
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];

export const run = async (tableName, movieYear1, movieTitle1) => {
    const params = {
        Statement: "DELETE FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
    };
    try {
        await ddbDocClient.send(new ExecuteStatementCommand(params));
        console.log("Success. Item deleted.");
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.error(err);
    }
};
run(tableName, movieYear1, movieTitle1);
```

Delete an item from a table by batch using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[5];
const movieTitle2 = process.argv[6];

export const run = async (
    tableName,
    movieYear1,
    movieTitle1,
    movieYear2,
    movieTitle2
) => {
    try {
        const params = {
            Statements: [
                {
                    Statement: "DELETE FROM " + tableName + " where year=? and title=?",
                    Parameters: [{ N: movieYear1 }, { S: movieTitle1 }],
                },
                {
                    Statement: "DELETE FROM " + tableName + " where year=? and title=?",
                    Parameters: [{ N: movieYear2 }, { S: movieTitle2 }],
                },
            ],
        };
        const data = await ddbDocClient.send(
            new BatchExecuteStatementCommand(params)
        );
        console.log("Success. Items deleted.", data);
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.error(err);
    }
};
run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [DeleteItem](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an item from a table.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
  TableName: 'TABLE',
  Key: {
    'KEY_NAME': {N: 'VALUE'}
  }
};

// Call DynamoDB to delete the item from the table
ddb.deleteItem(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

Delete an item from a table using the DynamoDB document client.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({apiVersion: '2012-08-10'});

var params = {
  Key: {
    'HASH_KEY': VALUE
  },
  TableName: 'TABLE'
};

docClient.delete(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteItem](#) in *AWS SDK for JavaScript API Reference*.

## Get a batch of items

The following code example shows how to get a batch of DynamoDB items.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create an Amazon DynamoDB service client object.  
const ddbClient = new DynamoDBClient({ region: REGION });  
export { ddbClient };
```

Get the items.

```
// Import required AWS SDK clients and commands for Node.js  
import { BatchGetItemCommand } from "@aws-sdk/client-dynamodb";  
import { ddbClient } from "./libs/ddbClient.js";  
  
// Set the parameters  
export const params = {  
    RequestItems: {  
        TABLE_NAME: {  
            Keys: [  
                {  
                    KEY_NAME_1: { N: "KEY_VALUE" },  
                    KEY_NAME_2: { N: "KEY_VALUE" },  
                    KEY_NAME_3: { N: "KEY_VALUE" },  
                },  
            ],  
            ProjectionExpression: "ATTRIBUTE_NAME",  
        },  
    },  
};  
  
export const run = async () => {  
    try {  
        const data = await ddbClient.send(new BatchGetItemCommand(params));  
        console.log("Success, items retrieved", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [BatchGetItem](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
  RequestItems: [
    {
      'TABLE_NAME': {
        Keys: [
          {'KEY_NAME': {N: 'KEY_VALUE_1'}},
          {'KEY_NAME': {N: 'KEY_VALUE_2'}},
          {'KEY_NAME': {N: 'KEY_VALUE_3'}}
        ],
        ProjectionExpression: 'KEY_NAME, ATTRIBUTE'
      }
    }
  ];
};

ddb.batchGetItem(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    data.Responses.TABLE_NAME.forEach(function(element, index, array) {
      console.log(element);
    });
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [BatchGetItem](#) in [AWS SDK for JavaScript API Reference](#).

## Get an item from a table

The following code example shows how to get an item from a DynamoDB table.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";
```

```
const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: true, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {
    marshallOptions,
    unmarshallOptions,
});

export { ddbDocClient };
```

Get an item from a table.

```
import { GetCommand } from "@aws-sdk/lib-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

// Set the parameters.
export const params = {
    TableName: "TABLE_NAME",
    Key: {
        primaryKey: "VALUE_1",
        sortKey: "VALUE_2",
    },
};

export const getItem = async () => {
    try {
        const data = await ddbDocClient.send(new GetCommand(params));
        console.log("Success :", data.Item);
    } catch (err) {
        console.log("Error", err);
    }
};
getItem();
```

Get an item from a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient";

const tableName = process.argv[2];
const movieTitle1 = process.argv[3];

export const run = async (tableName, movieTitle1) => {
    const params = {
        Statement: "SELECT * FROM " + tableName + " where title=?",
        ExpressionAttributeValues: {
            ":t": movieTitle1,
        },
    };
    const data = await ddbDocClient.send(new ExecuteStatementCommand(params));
    console.log(data.Items);
};
```

```
    Parameters: [{ S: movieTitle1 }],
};

try {
  const data = await ddbDocClient.send(new ExecuteStatementCommand(params));
  for (let i = 0; i < data.Items.length; i++) {
    console.log(
      "Success. The query return the following data. Item " + i,
      data.Items[i].year,
      data.Items[i].title,
      data.Items[i].info
    );
  }
  return "Run successfully"; // For unit tests.
} catch (err) {
  console.error(err);
}
};

run(tableName, movieTitle1);
```

Get items by batch from a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[5];
const movieTitle2 = process.argv[6];

export const run = async (
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2
) => {
  const params = {
    Statements: [
      {
        Statement: "SELECT * FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
      },
      {
        Statement: "SELECT * FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
      },
    ],
  };
  try {
    const data = await ddbDocClient.send(
      new BatchExecuteStatementCommand(params)
    );
    for (let i = 0; i < data.Responses.length; i++) {
      console.log(data.Responses[i].Item.year);
      console.log(data.Responses[i].Item.title);
    }
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.error(err);
  }
};
```

```
run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

- For API details, see [GetItem](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get an item from a table.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
  TableName: 'TABLE',
  Key: {
    'KEY_NAME': {N: '001'}
  },
  ProjectionExpression: 'ATTRIBUTE_NAME'
};

// Call DynamoDB to read the item from the table
ddb.getItem(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Item);
  }
});
```

Get an item from a table using the DynamoDB document client.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({apiVersion: '2012-08-10'});

var params = {
  TableName: 'EPISODES_TABLE',
  Key: {'KEY_NAME': VALUE}
};

docClient.get(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Item);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [GetItem](#) in *AWS SDK for JavaScript API Reference*.

## Get information about a table

The following code example shows how to get information about a DynamoDB table.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create an Amazon DynamoDB service client object.  
const ddbClient = new DynamoDBClient({ region: REGION });  
export { ddbClient };
```

Describe the table.

```
// Import required AWS SDK clients and commands for Node.js  
import { DescribeTableCommand } from "@aws-sdk/client-dynamodb";  
import { ddbClient } from "./libs/ddbClient.js";  
  
// Set the parameters  
export const params = { TableName: "TABLE_NAME" }; //TABLE_NAME  
  
export const run = async () => {  
    try {  
        const data = await ddbClient.send(new DescribeTableCommand(params));  
        console.log("Success", data);  
        // console.log("Success", data.Table.KeySchema);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeTable](#) in *AWS SDK for JavaScript API Reference*.

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create the DynamoDB service object  
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});
```

```
var params = {
  TableName: process.argv[2]
};

// Call DynamoDB to retrieve the selected table descriptions
ddb.describeTable(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Table.KeySchema);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DescribeTable](#) in [AWS SDK for JavaScript API Reference](#).

## List tables

The following code example shows how to list DynamoDB tables.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon DynamoDB service client object.
const ddbClient = new DynamoDBClient({ region: REGION });
export { ddbClient };
```

List the tables.

```
// Import required AWS SDK clients and commands for Node.js
import { ListTablesCommand } from "@aws-sdk/client-dynamodb";
import { ddbClient } from "./libs/ddbClient.js";

export const run = async () => {
  try {
    const data = await ddbClient.send(new ListTablesCommand({}));
    console.log(data);
    // console.log(data.TableNames.join("\n"));
    return data;
  } catch (err) {
    console.error(err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListTables](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

// Call DynamoDB to retrieve the list of tables
ddb.listTables({Limit: 10}, function(err, data) {
  if (err) {
    console.log("Error", err.code);
  } else {
    console.log("Table names are ", data.TableNames);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListTables](#) in *AWS SDK for JavaScript API Reference*.

## Put an item in a table

The following code example shows how to put an item in a DynamoDB table.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
  // Whether to automatically convert empty strings, blobs, and sets to `null`.
  convertEmptyValues: false, // false, by default.
  // Whether to remove undefined values while marshalling.
  removeUndefinedValues: true, // false, by default.
  // Whether to convert typeof object to map attribute.
  convertClassInstanceToMap: false, // false, by default.
};
```

```
const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {
    marshallOptions,
    unmarshallOptions,
});

export { ddbDocClient };
```

Put an item in a table using the DynamoDB document client.

```
import { PutCommand } from "@aws-sdk/lib-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

export const putItem = async () => {
    // Set the parameters.
    export const params = {
        TableName: "TABLE_NAME",
        Item: {
            primaryKey: "VALUE_1",
            sortKey: "VALUE_2",
        },
    };
    try {
        const data = await ddbDocClient.send(new PutCommand(params));
        console.log("Success - item added or updated", data);
    } catch (err) {
        console.log("Error", err.stack);
    }
};
putItem();
```

Put an item in a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];

export const run = async (tableName, movieTitle1, movieYear1) => {
    const params = {
        Statement: "INSERT INTO " + tableName + " value {'title':?:, 'year':?}",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
    };
    try {
        await ddbDocClient.send(new ExecuteStatementCommand(params));
        console.log("Success. Item added.");
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.error(err);
    }
};
```

```
run(tableName, movieTitle1, movieYear1);
```

Put items by batch in a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[5];
const movieTitle2 = process.argv[6];

export const run = async (
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2
) => {
  const params = {
    Statements: [
      {
        Statement:
          "INSERT INTO " + tableName + " value {'title':?, 'year':?}",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
      },
      {
        Statement:
          "INSERT INTO " + tableName + " value {'title':?, 'year':?}",
        Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
      },
    ],
  };
  try {
    await ddbDocClient.send(
      new BatchExecuteStatementCommand(params)
    );
    console.log("Success. Items added.");
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.error(err);
  }
};
run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

- For API details, see [PutItem](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Put an item in a table.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});
```

```
// Create the DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
  TableName: 'CUSTOMER_LIST',
  Item: {
    'CUSTOMER_ID' : {N: '001'},
    'CUSTOMER_NAME' : {S: 'Richard Roe'}
  }
};

// Call DynamoDB to add the item to the table
ddb.putItem(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

Put an item in a table using the DynamoDB document client.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({apiVersion: '2012-08-10'});

var params = {
  TableName: 'TABLE',
  Item: {
    'HASHKEY': VALUE,
    'ATTRIBUTE_1': 'STRING_VALUE',
    'ATTRIBUTE_2': VALUE_2
  }
};

docClient.put(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutItem](#) in [AWS SDK for JavaScript API Reference](#).

## Query a table

The following code example shows how to query a DynamoDB table.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create an Amazon DynamoDB service client object.  
const ddbClient = new DynamoDBClient({ region: REGION });  
export { ddbClient };
```

Query the table.

```
// Import required AWS SDK clients and commands for Node.js  
import { QueryCommand } from "@aws-sdk/client-dynamodb";  
import { ddbClient } from "./libs/ddbClient.js";  
  
// Set the parameters  
export const params = {  
    KeyConditionExpression: "Season = :s and Episode > :e",  
    FilterExpression: "contains (Subtitle, :topic)",  
    ExpressionAttributeValues: {  
        ":s": { N: "1" },  
        ":e": { N: "2" },  
        ":topic": { S: "SubTitle" },  
    },  
    ProjectionExpression: "Episode, Title, Subtitle",  
    TableName: "EPISODES_TABLE",  
};  
  
export const run = async () => {  
    try {  
        const data = await ddbClient.send(new QueryCommand(params));  
        data.Items.forEach(function (element) {  
            console.log(element.Title.S + " (" + element.Subtitle.S + ")");  
        });  
        return data;  
    } catch (err) {  
        console.error(err);  
    }  
};  
run();
```

Create the client for the DynamoDB document client.

```
// Create service client module using ES6 syntax.  
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";  
import { ddbClient } from "./ddbClient";  
  
const marshallOptions = {  
    // Whether to automatically convert empty strings, blobs, and sets to `null`.  
    convertEmptyValues: false, // false, by default.  
    // Whether to remove undefined values while marshalling.  
    removeUndefinedValues: false, // false, by default.  
    // Whether to convert typeof object to map attribute.  
    convertClassInstanceToMap: false, // false, by default.  
};  
  
const unmarshallOptions = {  
    // Whether to return numbers as a string instead of converting them to native  
    // JavaScript numbers.  
};
```

```
        wrapNumbers: false, // false, by default.  
};  
  
const translateConfig = { marshallOptions, unmarshallOptions };  
  
// Create the DynamoDB Document client.  
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, translateConfig);  
  
export { ddbDocClient };
```

Query the table using the DynamoDB document client.

```
import { QueryCommand } from "@aws-sdk/lib-dynamodb";  
import { ddbDocClient } from "../libs/ddbDocClient.js";  
  
// Set the parameters.  
export const params = {  
    ExpressionAttributeNames: { "#r": "rank", "#y": "year" },  
    ProjectionExpression: "#r, #y, title",  
    TableName: "TABLE_NAME",  
    ExpressionAttributeValues: {  
        ":t": "MOVIE_NAME",  
        ":y": "MOVIE_YEAR",  
        ":r": "MOVIE_RANK",  
    },  
    KeyConditionExpression: "title = :t and #y = :y",  
    FilterExpression: "info.#r = :r",  
};  
  
export const queryTable = async () => {  
    try {  
        const data = await ddbDocClient.send(new QueryCommand(params));  
        for (let i = 0; i < data.Items.length; i++) {  
            console.log(  
                "Success. Items with rank of " +  
                "MOVIE_RANK" +  
                " include\n" +  
                "Year = " +  
                data.Items[i].year +  
                " Title = " +  
                data.Items[i].title  
            );  
        }  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
queryTable();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Query](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region
```

```
AWS.config.update({region: 'REGION'});

// Create DynamoDB document client
var docClient = new AWS.DynamoDB.DocumentClient({apiVersion: '2012-08-10'});

var params = {
    ExpressionAttributeValues: {
        ':s': 2,
        ':e': 9,
        ':topic': 'PHRASE'
    },
    KeyConditionExpression: 'Season = :s and Episode > :e',
    FilterExpression: 'contains (Subtitle, :topic)',
    TableName: 'EPISODES_TABLE'
};

docClient.query(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data.Items);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Query in AWS SDK for JavaScript API Reference](#).

## Run a PartiQL statement

The following code example shows how to run a PartiQL statement on a DynamoDB table.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
export const REGION = "eu-west-1"; // For example, "us-east-1".
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: false, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
```

```
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

const translateConfig = { marshallOptions, unmarshallOptions };

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, translateConfig);

export { ddbDocClient };
```

Create an item using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];

export const run = async (tableName, movieTitle1, movieYear1) => {
    const params = {
        Statement: "INSERT INTO " + tableName + " value {'title':?:, 'year':?}",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
    };
    try {
        await ddbDocClient.send(new ExecuteStatementCommand(params));
        console.log("Success. Item added.");
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.error(err);
    }
};
run(tableName, movieTitle1, movieYear1);
```

Get an item using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient";

const tableName = process.argv[2];
const movieTitle1 = process.argv[3];

export const run = async (tableName, movieTitle1) => {
    const params = {
        Statement: "SELECT * FROM " + tableName + " where title=?",
        Parameters: [{ S: movieTitle1 }],
    };
    try {
        const data = await ddbDocClient.send(new ExecuteStatementCommand(params));
        for (let i = 0; i < data.Items.length; i++) {
            console.log(
                "Success. The query return the following data. Item " + i,
            );
        }
    } catch (err) {
        console.error(err);
    }
};
```

```
        data.Items[i].year,
        data.Items[i].title,
        data.Items[i].info
    );
}
return "Run successfully"; // For unit tests.
} catch (err) {
    console.error(err);
}
};

run(tableName, movieTitle1);
```

Update an item using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const producer1 = process.argv[5];

export const run = async (tableName, movieYear1, movieTitle1, producer1) => {
    const params = {
        Statement:
            "UPDATE " + tableName + " SET Producer=? where title=? and year=?",
        Parameters: [{ S: producer1 }, { S: movieTitle1 }, { N: movieYear1 }],
    };
    try {
        await ddbDocClient.send(new ExecuteStatementCommand(params));
        console.log("Success. Item updated.");
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.error(err);
    }
};
run(tableName, movieYear1, movieTitle1, producer1);
```

Delete an item using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];

export const run = async (tableName, movieYear1, movieTitle1) => {
    const params = {
        Statement: "DELETE FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
    };
    try {
        await ddbDocClient.send(new ExecuteStatementCommand(params));
        console.log("Success. Item deleted.");
        return "Run successfully"; // For unit tests.
    } catch (err) {
```

```
        console.error(err);
    }
};

run(tableName, movieYear1, movieTitle1);
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for JavaScript API Reference*.

## Run batches of PartiQL statements

The following code example shows how to run batches of PartiQL statements on a DynamoDB table.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
export const REGION = "eu-west-1"; // For example, "us-east-1".
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: false, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

const translateConfig = { marshallOptions, unmarshallOptions };

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, translateConfig);

export { ddbDocClient };
```

Create a batch of items using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[5];
const movieTitle2 = process.argv[6];

export const run = async (
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2
) => {
  const params = {
    Statements: [
      {
        Statement:
          "INSERT INTO " + tableName + " value {'title':?:, 'year':?}",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
      },
      {
        Statement:
          "INSERT INTO " + tableName + " value {'title':?:, 'year':?}",
        Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
      },
    ],
  };
  try {
    await ddbDocClient.send(
      new BatchExecuteStatementCommand(params)
    );
    console.log("Success. Items added.");
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.error(err);
  }
};
run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

Get a batch of items using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[5];
const movieTitle2 = process.argv[6];

export const run = async (
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2
) => {
  const params = {
```

```
Statements: [
  {
    Statement: "SELECT * FROM " + tableName + " where title=? and year=?",
    Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
  },
  {
    Statement: "SELECT * FROM " + tableName + " where title=? and year=?",
    Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
  },
],
];
try {
  const data = await ddbDocClient.send(
    new BatchExecuteStatementCommand(params)
  );
  for (let i = 0; i < data.Responses.length; i++) {
    console.log(data.Responses[i].Item.year);
    console.log(data.Responses[i].Item.title);
  }
  return "Run successfully"; // For unit tests.
} catch (err) {
  console.error(err);
}
};
run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

Update a batch of items using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[6];
const movieTitle2 = process.argv[7];
const producer1 = process.argv[5];
const producer2 = process.argv[8];

export const run = async (
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2,
  producer1,
  producer2
) => {
  const params = {
    Statements: [
      {
        Statement:
          "UPDATE " + tableName + " SET producer=? where title=? and year=?",
        Parameters: [{ S: producer1 }, { S: movieTitle1 }, { N: movieYear1 }],
      },
      {
        Statement:
          "UPDATE " + tableName + " SET producer=? where title=? and year=?",
        Parameters: [{ S: producer2 }, { S: movieTitle2 }, { N: movieYear2 }],
      },
    ],
  };
};
```

```
try {
    await ddbDocClient.send(
        new BatchExecuteStatementCommand(params)
    );
    console.log("Success. Items updated.");
    return "Run successfully"; // For unit tests.
} catch (err) {
    console.error(err);
}
};

run(
    tableName,
    movieYear1,
    movieTitle1,
    movieYear2,
    movieTitle2,
    producer1,
    producer2
);
```

Delete a batch of items using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[5];
const movieTitle2 = process.argv[6];

export const run = async (
    tableName,
    movieYear1,
    movieTitle1,
    movieYear2,
    movieTitle2
) => {
    try {
        const params = {
            Statements: [
                {
                    Statement: "DELETE FROM " + tableName + " where year=? and title=?",
                    Parameters: [{ N: movieYear1 }, { S: movieTitle1 }],
                },
                {
                    Statement: "DELETE FROM " + tableName + " where year=? and title=?",
                    Parameters: [{ N: movieYear2 }, { S: movieTitle2 }],
                },
            ],
        };
        const data = await ddbDocClient.send(
            new BatchExecuteStatementCommand(params)
        );
        console.log("Success. Items deleted.", data);
        return "Run successfully"; // For unit tests.
    } catch (err) {
        console.error(err);
    }
};
run(tableName, movieYear1, movieTitle1, movieYear2, movieTitle2);
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for JavaScript API Reference*.

## Scan a table

The following code example shows how to scan a DynamoDB table.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: true, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {
    marshallOptions,
    unmarshallOptions,
});

export { ddbDocClient };
```

Scan a table using the DynamoDB document client.

```
// Import required AWS SDK clients and commands for Node.js.
import { ddbDocClient } from "../../libs/ddbDocClient.js";
import { ScanCommand } from "@aws-sdk/lib-dynamodb";
// Set the parameters.
export const params = {
    TableName: "TABLE_NAME",
```

```
ProjectionExpression: "#r, #y, title",
ExpressionAttributeNames: { "#r": "rank", "#y": "year" },
FilterExpression: "title = :t and #y = :y and info.#r = :r",
ExpressionAttributeValues: {
    ":r": "MOVIE_RANK",
    ":y": "MOVIE_YEAR",
    ":t": "MOVIE_NAME",
},
};

export const scanTable = async () => {
    try {
        const data = await ddbDocClient.send(new ScanCommand(params));
        console.log("success", data.Items);
    } catch (err) {
        console.log("Error", err);
    }
};
scanTable();
```

- For API details, see [Scan](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js.
var AWS = require("aws-sdk");
// Set the AWS Region.
AWS.config.update({ region: "REGION" });

// Create DynamoDB service object.
var ddb = new AWS.DynamoDB({ apiVersion: "2012-08-10" });

const params = {
    // Specify which items in the results are returned.
    FilterExpression: "Subtitle = :topic AND Season = :s AND Episode = :e",
    // Define the expression attribute value, which are substitutes for the values you
    want to compare.
    ExpressionAttributeValues: {
        ":topic": {S: "SubTitle2"},
        ":s": {N: 1},
        ":e": {N: 2},
    },
    // Set the projection expression, which are the attributes that you want.
    ProjectionExpression: "Season, Episode, Title, Subtitle",
    TableName: "EPISODES_TABLE",
};

ddb.scan(params, function (err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
        data.Items.forEach(function (element, index, array) {
            console.log(
                "printing",
                element.Title.S + " (" + element.Subtitle.S + ")"
            );
        });
    }
})
```

```
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Scan](#) in [AWS SDK for JavaScript API Reference](#).

## Update an item in a table

The following code example shows how to update an item in a DynamoDB table.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the DynamoDB document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: true, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {
    marshallOptions,
    unmarshallOptions,
});

export { ddbDocClient };
```

Update an item in a table using the DynamoDB document client.

```
import { UpdateCommand } from "@aws-sdk/lib-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";
```

```
export const updateItem = async () => {
  // Set the parameters.
  const params = {
    TableName: "TABLE_NAME",
    Key: {
      title: "MOVIE_NAME",
      year: "MOVIE_YEAR",
    },
    ProjectionExpression: "#r",
    ExpressionAttributeNames: { "#r": "rank" },
    UpdateExpression: "set info.plot = :p, info.#r = :r",
    ExpressionAttributeValues: {
      ":p": "MOVIE_PLOT",
      ":r": "MOVIE_RANK",
    },
  };
  try {
    const data = await ddbDocClient.send(new UpdateCommand(params));
    console.log("Success - item added or updated", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
updateItem();
```

Update an item in a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { ExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";

const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const producer1 = process.argv[5];

export const run = async (tableName, movieYear1, movieTitle1, producer1) => {
  const params = {
    Statement:
      "UPDATE " + tableName + " SET Producer=? where title=? and year=?",
    Parameters: [{ S: producer1 }, { S: movieTitle1 }, { N: movieYear1 }],
  };
  try {
    await ddbDocClient.send(new ExecuteStatementCommand(params));
    console.log("Success. Item updated.");
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.error(err);
  }
};
run(tableName, movieYear1, movieTitle1, producer1);
```

Update items by batch in a table using PartiQL.

```
// Import required AWS SDK clients and commands for Node.js.
import { BatchExecuteStatementCommand } from "@aws-sdk/client-dynamodb";
import { ddbDocClient } from "../libs/ddbDocClient.js";
```

```
const tableName = process.argv[2];
const movieYear1 = process.argv[3];
const movieTitle1 = process.argv[4];
const movieYear2 = process.argv[6];
const movieTitle2 = process.argv[7];
const producer1 = process.argv[5];
const producer2 = process.argv[8];

export const run = async (
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2,
  producer1,
  producer2
) => {
  const params = {
    Statements: [
      {
        Statement:
          "UPDATE " + tableName + " SET producer=? where title=? and year=?",
        Parameters: [{ S: producer1 }, { S: movieTitle1 }, { N: movieYear1 }],
      },
      {
        Statement:
          "UPDATE " + tableName + " SET producer=? where title=? and year=?",
        Parameters: [{ S: producer2 }, { S: movieTitle2 }, { N: movieYear2 }],
      },
    ],
  };
  try {
    await ddbDocClient.send(
      new BatchExecuteStatementCommand(params)
    );
    console.log("Success. Items updated.");
    return "Run successfully"; // For unit tests.
  } catch (err) {
    console.error(err);
  }
};
run(
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2,
  producer1,
  producer2
);
```

- For API details, see [UpdateItem](#) in *AWS SDK for JavaScript API Reference*.

## Write a batch of items

The following code example shows how to write a batch of DynamoDB items.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.  
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";  
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";  
// Create an Amazon DynamoDB service client object.  
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create the document client.

```
// Create a service client module using ES6 syntax.  
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";  
import { ddbClient } from "./ddbClient.js";  
  
const marshallOptions = {  
    // Whether to automatically convert empty strings, blobs, and sets to `null`.  
    convertEmptyValues: false, // false, by default.  
    // Whether to remove undefined values while marshalling.  
    removeUndefinedValues: true, // false, by default.  
    // Whether to convert typeof object to map attribute.  
    convertClassInstanceToMap: false, // false, by default.  
};  
  
const unmarshallOptions = {  
    // Whether to return numbers as a string instead of converting them to native  
    // JavaScript numbers.  
    wrapNumbers: false, // false, by default.  
};  
  
// Create the DynamoDB document client.  
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {  
    marshallOptions,  
    unmarshallOptions,  
});  
  
export { ddbDocClient };
```

Add the items to the table.

```
import fs from "fs";  
import * as R from "ramda";  
import { ddbDocClient } from "../libs/ddbDocClient.js";  
import { BatchWriteCommand } from "@aws-sdk/lib-dynamodb";  
  
export const writeData = async () => {  
    // Before you run this example, download 'movies.json' from https://  
    // docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Js.02.html,  
    // and put it in the same folder as the example.  
    // Get the movie data parse to convert into a JSON object.  
    const allMovies = JSON.parse(fs.readFileSync("moviedata.json", "utf8"));  
    // Split the table into segments of 25.  
    const dataSegments = R.splitEvery(25, allMovies);  
    const TABLE_NAME = "TABLE_NAME"  
    try {  
        // Loop batch write operation 10 times to upload 250 items.  
        for (let i = 0; i < 10; i++) {  
            const segment = dataSegments[i];  
            for (let j = 0; j < 25; j++) {  
                const params = {
```

```
RequestItems: {
    [TABLE_NAME]: [
        {
            // Destination Amazon DynamoDB table name.
            PutRequest: {
                Item: {
                    year: segment[j].year,
                    title: segment[j].title,
                    info: segment[j].info,
                },
            },
        ],
    ],
},
ddbDocClient.send(new BatchWriteCommand(params));
}
console.log("Success, table updated.");
}
} catch (error) {
    console.log("Error", error);
}
};
writeData();
}
```

- For API details, see [BatchWriteItem](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create DynamoDB service object
var ddb = new AWS.DynamoDB({apiVersion: '2012-08-10'});

var params = {
    RequestItems: {
        "TABLE_NAME": [
            {
                PutRequest: {
                    Item: {
                        "KEY": { "N": "KEY_VALUE" },
                        "ATTRIBUTE_1": { "S": "ATTRIBUTE_1_VALUE" },
                        "ATTRIBUTE_2": { "N": "ATTRIBUTE_2_VALUE" }
                    }
                }
            },
            {
                PutRequest: {
                    Item: {
                        "KEY": { "N": "KEY_VALUE" },
                        "ATTRIBUTE_1": { "S": "ATTRIBUTE_1_VALUE" },
                        "ATTRIBUTE_2": { "N": "ATTRIBUTE_2_VALUE" }
                    }
                }
            }
        ]
    }
}
```

```
        }
    });

ddb.batchWriteItem(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [BatchWriteItem in AWS SDK for JavaScript API Reference](#).

## Scenarios

### Get started using tables, items, and queries

The following code example shows how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.
- Delete a movie from the table.
- Delete the table.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a DynamoDB client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import { DEFAULT_REGION } from "../../../../../libs/utils/util-aws-sdk.js";
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: DEFAULT_REGION });
```

Create a DynamoDB document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: true, // false, by default.
    // Whether to convert typeof object to map attribute.
```

```
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
  // Whether to return numbers as a string instead of converting them to native
  // JavaScript numbers.
  wrapNumbers: false, // false, by default.
};

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, {
  marshallOptions,
  unmarshallOptions,
});

export { ddbDocClient };
```

Run the scenario.

```
import fs from "fs";
import { splitEvery } from "ramda";
import {
  PutCommand,
  GetCommand,
  UpdateCommand,
  BatchWriteCommand,
  DeleteCommand,
  ScanCommand,
  QueryCommand,
} from "@aws-sdk/lib-dynamodb";
import {
  CreateTableCommand,
  DeleteTableCommand,
  waitUntilTableExists,
  waitUntilTableNotExists,
} from "@aws-sdk/client-dynamodb";

import { ddbClient } from "../libs/ddbClient.js";
import { ddbDocClient } from "../libs/ddbDocClient.js";

/**
 * @param {string} tableName
 */
const createTable = async (tableName) => {
  await ddbClient.send(
    new CreateTableCommand({
      AttributeDefinitions: [
        {
         AttributeName: "year",
         AttributeType: "N",
        },
        {
         AttributeName: "title",
         AttributeType: "S",
        },
      ],
      KeySchema: [
        {
         AttributeName: "year",
         KeyType: "HASH",
        },
        {
         AttributeName: "title",
         KeyType: "RANGE",
        },
      ],
    })
  );
};

createTable("Books").catch((err) => {
  console.error(`Error creating table ${err}`);
});
```

```

        },
    ],
    // Enables "on-demand capacity mode".
    BillingMode: "PAY_PER_REQUEST",
    TableName: tableName,
  })
);
await waitUntilTableExists(
  { client: ddbClient, maxWaitTime: 15, maxDelay: 2, minDelay: 1 },
  { TableName: tableName }
);
};

/***
 *
 * @param {string} tableName
 * @param {Record<string, any> | undefined} attributes
 */
const putItem = async (tableName, attributes) => {
  const command = new PutCommand({
    TableName: tableName,
    Item: attributes,
  });

  await ddbDocClient.send(command);
};

/***
 *
 * @param {string} tableName
 * @param {string} filePath
 * @returns { { movieCount: number } } The number of movies written to the database.
 */
const batchWriteMoviesFromFile = async (tableName, filePath) => {
  const fileContents = fs.readFileSync(filePath);
  const movies = JSON.parse(fileContents, "utf8");

  // Map movies to RequestItems.
  const putMovieRequestItems = movies.map(({ year, title, info }) => ({
    PutRequest: { Item: { year, title, info } },
  }));

  // Organize RequestItems into batches of 25. 25 is the max number of items in a batch request.
  const putMovieBatches = splitEvery(25, putMovieRequestItems);
  const batchCount = putMovieBatches.length;

  // Map batches to promises.
  const batchRequests = putMovieBatches.map(async (batch, i) => {
    const command = new BatchWriteCommand({
      RequestItems: {
        [tableName]: batch,
      },
    });

    await ddbDocClient.send(command).then(() => {
      console.log(`Wrote batch ${i + 1} of ${batchCount} with ${batch.length} items.`);
    });
  });
};

// Wait for all batch requests to resolve.
await Promise.all(batchRequests);

return { movieCount: movies.length };

```

```
};

/**
 *
 * @param {string} tableName
 * @param {{
 *   existingMovieName: string,
 *   existingMovieYear: string,
 *   newMoviePlot: string,
 *   newMovieRank: string}} keyUpdate
 */
const updateMovie = async (
  tableName,
  { existingMovieName, existingMovieYear, newMoviePlot, newMovieRank }
) => {
  await ddbClient.send(
    new UpdateCommand({
      TableName: tableName,
      Key: {
        title: existingMovieName,
        year: existingMovieYear,
      },
      // Define expressions for the new or updated attributes.
      ExpressionAttributeNames: { "#r": "rank" },
      UpdateExpression: "set info.plot = :p, info.#r = :r",
      ExpressionAttributeValues: {
        ":p": newMoviePlot,
        ":r": newMovieRank,
      },
      ReturnValues: "ALL_NEW",
    })
  );
};

/**
 * @param {{ title: string, info: { plot: string, rank: number }, year: number }} movie
 */
const logMovie = (movie) => {
  console.log(` | Title: "${movie.title}"`);
  console.log(` | Plot: "${movie.info.plot}"`);
  console.log(` | Year: ${movie.year}`);
  console.log(` | Rank: ${movie.info.rank}`);
};

/**
 *
 * @param {{ title: string, info: { plot: string, rank: number }, year: number }[]} movies
 */
const logMovies = (movies) => {
  console.log("\n");
  movies.forEach((movie, i) => {
    if (i > 0) {
      console.log("-".repeat(80));
    }

    logMovie(movie);
  });
};

/**
 *
 * @param {string} tableName
 * @param {string} title
 * @param {number} year
 * @returns

```

```
/*
const getMovie = async (tableName, title, year) => {
    const { Item } = await ddbDocClient.send(
        new GetCommand({
            TableName: tableName,
            Key: {
                title,
                year,
            },
            // By default, reads are eventually consistent. "ConsistentRead: true" represents
            // a strongly consistent read. This guarantees that the most up-to-date data is
            // returned. It
            // can also result in higher latency and a potential for server errors.
            ConsistentRead: true,
        })
    );
    return Item;
};

/**
 *
 * @param {string} tableName
 * @param {{ title: string, year: number }} key
 */
const deleteMovie = async (tableName, key) => {
    await ddbDocClient.send(
        new DeleteCommand({
            TableName: tableName,
            Key: key,
        })
    );
};

/**
 *
 * @param {string} tableName
 * @param {number} startYear
 * @param {number} endYear
 * @param {Record<string, any>} startKey
 * @returns {Promise<{}[]>}
 */
const findMoviesBetweenYears = async (
    tableName,
    startYear,
    endYear,
    startKey = undefined
) => {
    const { Items, LastEvaluatedKey } = await ddbClient.send(
        new ScanCommand({
            ConsistentRead: true,
            TableName: tableName,
            ExpressionAttributeNames: { "#y": "year" },
            FilterExpression: "#y BETWEEN :y1 AND :y2",
            ExpressionAttributeValues: { ":y1": startYear, ":y2": endYear },
            ExclusiveStartKey: startKey,
        })
    );
    if (LastEvaluatedKey) {
        return Items.concat(
            await findMoviesBetweenYears(
                tableName,
                startYear,
                endYear,
                LastEvaluatedKey
            )
        );
    }
};
```

```
        )
    );
} else {
    return Items;
}
};

/***
 *
 * @param {string} tableName
 * @param {number} year
 * @returns
 */
const queryMoviesByYear = async (tableName, year) => {
    const command = new QueryCommand({
        ConsistentRead: true,
        ExpressionAttributeNames: { "#y": "year" },
        TableName: tableName,
        ExpressionAttributeValues: {
            ":y": year,
        },
        KeyConditionExpression: "#y = :y",
    });
    const { Items } = await ddbDocClient.send(command);
    return Items;
};

/***
 *
 * @param {*} tableName
 */
const deleteTable = async (tableName) => {
    await ddbDocClient.send(new DeleteTableCommand({ TableName: tableName }));
    await waitUntilTableNotExists(
        {
            client: ddbClient,
            maxWaitTime: 10,
            maxDelay: 2,
            minDelay: 1,
        },
        { TableName: tableName }
    );
};

export const runScenario = async ({
    tableName,
    newMovieName,
    newMovieYear,
    existingMovieName,
    existingMovieYear,
    newMovieRank,
    newMoviePlot,
    moviesPath,
}) => {
    console.log(`Creating table named: ${tableName}`);
    await createTable(tableName);
    console.log(`\nTable created.`);

    console.log(`\nAdding "${newMovieName}" to ${tableName}.`);
    await putItem(tableName, { title: newMovieName, year: newMovieYear });
    console.log("\nSuccess - single movie added.");

    console.log("\nWriting hundreds of movies in batches.");
    const { movieCount } = await batchWriteMoviesFromFile(tableName, moviesPath);
```

```
console.log(`\nWrote ${movieCount} movies to database.`);

console.log(`\nGetting "${existingMovieName}"`);
const originalMovie = await getMovie(
  tableName,
  existingMovieName,
  existingMovieYear
);
logMovie(originalMovie);

console.log(`\nUpdating "${existingMovieName}" with a new plot and rank.`);
await updateMovie(tableName, {
  existingMovieName,
  existingMovieYear,
  newMoviePlot,
  newMovieRank,
});
console.log(`\n"${existingMovieName}" updated.`);

console.log(`\nGetting latest info for "${existingMovieName}"`);
const updatedMovie = await getMovie(
  tableName,
  existingMovieName,
  existingMovieYear
);
logMovie(updatedMovie);

console.log(`\nDeleting "${newMovieName}"`);
await deleteMovie(tableName, { title: newMovieName, year: newMovieYear });
console.log(`\n"${newMovieName} deleted.`);

const [scanY1, scanY2] = [1985, 2003];
console.log(
  `\nScanning ${tableName} for movies that premiered between ${scanY1} and
${scanY2}.`;
);
const scannedMovies = await findMoviesBetweenYears(tableName, scanY1, scanY1);
logMovies(scannedMovies);

const queryY = 2003;
console.log(`Querying ${tableName} for movies that premiered in ${queryY}.`);
const queriedMovies = await queryMoviesByYear(tableName, queryY);
logMovies(queriedMovies);

console.log(`Deleting ${tableName}.`);
await deleteTable(tableName);
console.log(`${tableName} deleted.`);
};

const main = async () => {
  const args = {
    tableName: "myNewTable",
    newMovieName: "myMovieName",
    newMovieYear: 2022,
    existingMovieName: "This Is the End",
    existingMovieYear: 2013,
    newMovieRank: 200,
    newMoviePlot: "A coder cracks code...",
    moviesPath: "../../../../../resources/sample_files/movies.json",
  };
  try {
    await runScenario(args);
  } catch (err) {
    // Some extra error handling here to be sure the table is cleaned up if something
    // goes wrong during the scenario run.
  }
};
```

```
    console.error(err);

    const tableName = args.tableName;

    if (tableName) {
        console.log(`Attempting to delete ${tableName}`);
        await ddbClient
            .send(new DeleteTableCommand({ TableName: tableName }))
            .then(() => console.log(`\n${tableName} deleted.`))
            .catch((err) => console.error(`\nFailed to delete ${tableName}.`, err));
    }
};

export { main };
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.

- [BatchWriteItem](#)
- [CreateTable](#)
- [DeleteItem](#)
- [DeleteTable](#)
- [DescribeTable](#)
- [GetItem](#)
- [PutItem](#)
- [Query](#)
- [Scan](#)
- [UpdateItem](#)

## Query a table by using batches of PartiQL statements

The following code example shows how to:

- Get a batch of items by running multiple SELECT statements.
- Add a batch of items by running multiple INSERT statements.
- Update a batch of items by running multiple UPDATE statements.
- Delete a batch of items by running multiple DELETE statements.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
export const REGION = "eu-west-1"; // For example, "us-east-1".
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: REGION });
```

Create the document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
    // Whether to automatically convert empty strings, blobs, and sets to `null`.
    convertEmptyValues: false, // false, by default.
    // Whether to remove undefined values while marshalling.
    removeUndefinedValues: false, // false, by default.
    // Whether to convert typeof object to map attribute.
    convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
    // Whether to return numbers as a string instead of converting them to native
    // JavaScript numbers.
    wrapNumbers: false, // false, by default.
};

const translateConfig = { marshallOptions, unmarshallOptions };

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, translateConfig);

export { ddbDocClient };
```

Query items by batch.

```
/*
import fs from "fs";
// A practical functional library used to split the data into segments.
import * as R from "ramda";
import { ddbClient } from "../libs/ddbClient.js";
import { ddbDocClient } from "../libs/ddbDocClient.js";
import { BatchWriteCommand } from "@aws-sdk/lib-dynamodb";
import {
    CreateTableCommand,
    BatchExecuteStatementCommand,
} from "@aws-sdk/client-dynamodb";
if (process.argv.length < 6) {
    console.log(
        "Usage: node partiQL_basics.js <tableName> <movieTitle1> <movieYear1> <movieTitle1>
        <movieYear1> <producer1> <producer2> \n" +
        "Example: node partiQL_basics.js Movies_batch 2006 'The Departed' 2013 '2 Guns'
        'New View Films' 'Old Thyme Films'"
    );
}

const tableName = process.argv[2];
const movieYear1 = parseInt(process.argv[3]);
const movieTitle1 = process.argv[4];
const movieYear2 = parseInt(process.argv[5]);
const movieTitle2 = process.argv[6];
const producer1 = process.argv[7];
const producer2 = process.argv[8];

// Helper function to delay running the code while the AWS service calls wait for
// responses.
function wait(ms) {
    var start = Date.now();
    var end = start;
    while (end < start + ms) {
```

```
        end = Date.now();
    }
}
// Set the parameters.

export const run = async (
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2,
  producer1,
  producer2
) => {
  try {
    console.log("Creating table ...");
    // Set the parameters.
    const params = {
      AttributeDefinitions: [
        {
          AttributeName: "title",
          AttributeType: "S",
        },
        {
          AttributeName: "year",
          AttributeType: "N",
        },
      ],
      KeySchema: [
        {
          AttributeName: "title",
          KeyType: "HASH",
        },
        {
          AttributeName: "year",
          KeyType: "RANGE",
        },
      ],
      ProvisionedThroughput: {
        ReadCapacityUnits: 5,
        WriteCapacityUnits: 5,
      },
      TableName: tableName,
    };
    const data = await ddbClient.send(new CreateTableCommand(params));
    console.log("Waiting for table to be created...");
    wait(10000);
    console.log(
      "Table created. Table name is ",
      data.TableDescription.TableName
    );
    try {
      // Before you run this example, download 'movies.json' from https://
      docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Js.02.html,
      // and put it in the same folder as the example.
      // Get the movie data parse to convert into a JSON object.
      const allMovies = JSON.parse(fs.readFileSync("moviedata.json", "utf8"));
      // Split the table into segments of 25.
      const dataSegments = R.splitEvery(25, allMovies);
      // Loop batch write operation 10 times to upload 250 items.
      console.log("Writing movies in batch to table...");
      for (let i = 0; i < 10; i++) {
        const segment = dataSegments[i];
        for (let j = 0; j < 25; j++) {
          const params = {
            RequestItems: {
```

```
[tableName]: [
  {
    // Destination Amazon DynamoDB table name.
    PutRequest: {
      Item: {
        year: segment[j].year,
        title: segment[j].title,
        info: segment[j].info,
      },
    },
  },
],
};

ddbDocClient.send(new BatchWriteCommand(params));
}

}

wait(10000);
console.log("Success, movies written to table.");
try {
  console.log("Getting movie....");
  const params = {
    Statements: [
      {
        Statement:
          "SELECT * FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
      },
      {
        Statement:
          "SELECT * FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
      },
    ],
  };
  const data = await ddbDocClient.send(
    new BatchExecuteStatementCommand(params)
  );
  console.log("Success. The query return the following data.", data);
  for (let i = 0; i < data.Responses.length; i++) {
    console.log(data.Responses[i].Item.year);
    console.log(data.Responses[i].Item.title);
  }
}
try {
  const params = {
    Statements: [
      {
        Statement:
          "DELETE FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
      },
      {
        Statement:
          "DELETE FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
      },
    ],
  };
  await ddbDocClient.send(
    new BatchExecuteStatementCommand(params)
  );
  console.log("Success. Items deleted by batch.");
}
try {
  const params = {
    Statements: [
      {
        Statement:
          "DELETE FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
      },
      {
        Statement:
          "DELETE FROM " + tableName + " where title=? and year=?",
        Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],
      },
    ],
  };
  await ddbDocClient.send(
    new BatchExecuteStatementCommand(params)
  );
}
```

```
Statement:  
    "INSERT INTO " +  
    tableName +  
    " value {'title':?, 'year':?}"  
Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],  
,  
{  
    Statement:  
        "INSERT INTO " +  
        tableName +  
        " value {'title':?, 'year':?}"  
    Parameters: [{ S: movieTitle2 }, { N: movieYear2 }],  
,  
]  
};  
await ddbDocClient.send(  
    new BatchExecuteStatementCommand(params)  
);  
console.log("Success. Items added by batch.");  
try {  
    const params = {  
        Statements: [  
            {  
                Statement:  
                    "UPDATE " +  
                    tableName +  
                    " SET Producer=? where title=? and year=?"  
                Parameters: [  
                    { S: producer1 },  
                    { S: movieTitle1 },  
                    { N: movieYear1 }  
                ],  
            },  
            {  
                Statement:  
                    "UPDATE " +  
                    tableName +  
                    " SET Producer=? where title=? and year=?"  
                Parameters: [  
                    { S: producer2 },  
                    { S: movieTitle2 },  
                    { N: movieYear2 }  
                ],  
            },  
        ],  
    };  
    console.log("Updating movies...");  
    await ddbDocClient.send(  
        new BatchExecuteStatementCommand(params)  
    );  
    console.log("Success. Items updated by batch.");  
    return "Run successfully"; // For unit tests.  
} catch (err) {  
    console.log("Error updating items by batch. ", err);  
}  
} catch (err) {  
    console.log("Error adding items to table by batch. ", err);  
}  
} catch (err) {  
    console.log("Error deleting movies by batch. ", err);  
}  
} catch (err) {  
    console.log("Error getting movies by batch. ", err);  
}  
} catch (err) {  
    console.log("Error adding movies by batch. ", err);  
}
```

```
        }
    } catch (err) {
    console.log("Error creating table. ", err);
}
};

run(
  tableName,
  movieYear1,
  movieTitle1,
  movieYear2,
  movieTitle2,
  producer1,
  producer2
);
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for JavaScript API Reference*.

## Query a table using PartiQL

The following code example shows how to:

- Get an item by running a SELECT statement.
- Add an item by running an INSERT statement.
- Update an item by running an UPDATE statement.
- Delete an item by running a DELETE statement.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create the DynamoDB service client module using ES6 syntax.
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
// Set the AWS Region.
export const REGION = "eu-west-1"; // For example, "us-east-1".
// Create an Amazon DynamoDB service client object.
export const ddbClient = new DynamoDBClient({ region: REGION });
```

Create the document client.

```
// Create a service client module using ES6 syntax.
import { DynamoDBDocumentClient } from "@aws-sdk/lib-dynamodb";
import { ddbClient } from "./ddbClient.js";

const marshallOptions = {
  // Whether to automatically convert empty strings, blobs, and sets to `null`.
  convertEmptyValues: false, // false, by default.
  // Whether to remove undefined values while marshalling.
  removeUndefinedValues: false, // false, by default.
  // Whether to convert typeof object to map attribute.
  convertClassInstanceToMap: false, // false, by default.
};

const unmarshallOptions = {
```

```
// Whether to return numbers as a string instead of converting them to native
// JavaScript numbers.
wrapNumbers: false, // false, by default.
};

const translateConfig = { marshallOptions, unmarshallOptions };

// Create the DynamoDB document client.
const ddbDocClient = DynamoDBDocumentClient.from(ddbClient, translateConfig);

export { ddbDocClient };
```

### Query single items.

```
/*
import fs from "fs";
// A practical functional library used to split the data into segments.
import * as R from "ramda";
import { ddbClient } from "../libs/ddbClient.js";
import { ddbDocClient } from "../libs/ddbDocClient.js";
import { BatchWriteCommand } from "@aws-sdk/lib-dynamodb";
import {
  CreateTableCommand,
  ExecuteStatementCommand,
} from "@aws-sdk/client-dynamodb";
if (process.argv.length < 6) {
  console.log(
    "Usage: node partiQL_basics.js <tableName> <movieYear1> <movieTitle1> <producer1>
\n" +
    "Example: node partiQL_basics.js Movies 2006 'The Departed' 'New View Films'"
  );
}

// Helper function to delay running the code while the AWS service calls wait for
responses.
function wait(ms) {
  var start = Date.now();
  var end = start;
  while (end < start + ms) {
    end = Date.now();
  }
}

const tableName = process.argv[2];
const movieTitle1 = process.argv[3];
const movieYear1 = process.argv[4];
const producer1 = process.argv[5];

export const run = async (tableName, movieYear1, movieTitle1, producer1) => {
  try {
    console.log("Creating table ...");
    // Set the parameters.
    const params = {
      AttributeDefinitions: [
        {
          AttributeName: "title",
          AttributeType: "S",
        },
        {
          AttributeName: "year",
          AttributeType: "N",
        },
      ],
      KeySchema: [
```

```
{
  AttributeName: "title",
  KeyType: "HASH",
},
{
  AttributeName: "year",
  KeyType: "RANGE",
},
],
ProvisionedThroughput: {
  ReadCapacityUnits: 5,
  WriteCapacityUnits: 5,
},
TableName: tableName,
};

const data = await ddbClient.send(new CreateTableCommand(params));
console.log("Waiting for table to be created...");
wait(10000);
console.log(
  "Table created. Table name is ",
  data.TableDescription.TableName
);
try {
  // Before you run this example, download 'movies.json' from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Js.02.html,
  // and put it in the same folder as the example.
  // Get the movie data parse to convert into a JSON object.
  const allMovies = JSON.parse(fs.readFileSync("moviedata.json", "utf8"));
  // Split the table into segments of 25.
  const dataSegments = R.splitEvery(25, allMovies);
  // Loop batch write operation 10 times to upload 250 items.
  console.log("Writing movies in batch to table...");
  for (let i = 0; i < 10; i++) {
    const segment = dataSegments[i];
    for (let j = 0; j < 25; j++) {
      const params = {
        RequestItems: [
          [tableName]: [
            {
              // Destination Amazon DynamoDB table name.
              PutRequest: {
                Item: {
                  year: segment[j].year,
                  title: segment[j].title,
                  info: segment[j].info,
                },
              },
            ],
          ],
        ];
      };
      ddbDocClient.send(new BatchWriteCommand(params));
    }
  }
  wait(20000);
  console.log("Success, movies written to table.");
  try {
    const params = {
      Statement: "SELECT * FROM " + tableName + " where title=?",
      Parameters: [{ S: movieTitle1 }],
    };
    console.log("Getting movie....");

    console.log("Statement", params.Statement);
    const data = await ddbDocClient.send(
      new ExecuteStatementCommand(params)
    );
  }
}
```

```
    );
    for (let i = 0; i < data.Items.length; i++) {
        console.log(
            "Success. The query return the following data. Item " + i,
            data.Items[i].year,
            data.Items[i].title,
            data.Items[i].info
        );
    }
    try {
        const params = {
            Statement: "DELETE FROM " + tableName + " where title=? and year=?",
            Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
        };
        await ddbDocClient.send(
            new ExecuteStatementCommand(params)
        );
        console.log("Success. Item deleted.");
        try {
            const params = {
                Statement:
                    "INSERT INTO " + tableName + " value {'title':?, 'year':?}",
                Parameters: [{ S: movieTitle1 }, { N: movieYear1 }],
            };
            await ddbDocClient.send(
                new ExecuteStatementCommand(params)
            );
            console.log("Success. Item added.");
            try {
                const params = {
                    Statement:
                        "UPDATE " +
                        tableName +
                        " SET Producer=? where title=? and year=?",
                    Parameters: [
                        { S: producer1 },
                        { S: movieTitle1 },
                        { N: movieYear1 },
                    ],
                };
                console.log("Updating a single movie...");
                await ddbDocClient.send(
                    new ExecuteStatementCommand(params)
                );
                console.log("Success. Item updated.");
                return "Run successfully"; // For unit tests.
            } catch (err) {
                console.log("Error updating item. ", err);
            }
        } catch (err) {
            console.log("Error adding items to table. ", err);
        }
    } catch (err) {
        console.log("Error deleting movie. ", err);
    }
} catch (err) {
    console.log("Error getting movie. ", err);
}
} catch (err) {
    console.log("Error adding movies by batch. ", err);
}
} catch (err) {
    console.log("Error creating table. ", err);
}
};
```

```
run(tableName, movieYear1, movieTitle1, producer1);
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for JavaScript API Reference*.

## EventBridge examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon EventBridge.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2993\)](#)

## Actions

### Add a Lambda function target

The following code example shows how to add an AWS Lambda function target to an Amazon EventBridge event.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { EventBridgeClient } from "@aws-sdk/client-eventbridge";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon EventBridge service client object.
export const ebClient = new EventBridgeClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js.
import { PutTargetsCommand } from "@aws-sdk/client-eventbridge";
import { ebClient } from "./libs/eventBridgeClient.js";

// Set the parameters.
export const params = {
  Rule: "DEMO_EVENT",
  Targets: [
    {
      Arn: "LAMBDA_FUNCTION_ARN", //LAMBDA_FUNCTION_ARN
      Id: "myCloudWatchEventsTarget",
    },
  ],
};
```

```
export const run = async () => {
  try {
    const data = await ebClient.send(new PutTargetsCommand(params));
    console.log("Success, target added; requestID: ", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For API details, see [PutTargets](#) in *AWS SDK for JavaScript API Reference*.

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({apiVersion: '2015-10-07'});

var params = {
  Rule: 'DEMO_EVENT',
  Targets: [
    {
      Arn: 'LAMBDA_FUNCTION_ARN',
      Id: 'myEventBridgeTarget',
    }
  ]
};

ebevents.putTargets(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For API details, see [PutTargets](#) in *AWS SDK for JavaScript API Reference*.

### Create a scheduled rule

The following code example shows how to create an Amazon EventBridge scheduled rule.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { EventBridgeClient } from "@aws-sdk/client-eventbridge";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon EventBridge service client object.
export const ebClient = new EventBridgeClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js.
import { PutRuleCommand } from "@aws-sdk/client-eventbridge";
import { ebClient } from "./libs/eventBridgeClient.js";

// Set the parameters.
export const params = {
  Name: "DEMO_EVENT",
  RoleArn: "IAM_ROLE_ARN", //IAM_ROLE_ARN
  ScheduleExpression: "rate(5 minutes)",
  State: "ENABLED",
};

export const run = async () => {
  try {
    const data = await ebClient.send(new PutRuleCommand(params));
    console.log("Success, scheduled rule created; Rule ARN:", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For API details, see [PutRule](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({apiVersion: '2015-10-07'});

var params = {
  Name: 'DEMO_EVENT',
  RoleArn: 'IAM_ROLE_ARN',
  ScheduleExpression: 'rate(5 minutes)',
  State: 'ENABLED'
};

ebevents.putRule(params, function(err, data) {
  if (err) {
    console.log("Error", err);
```

```
    } else {
      console.log("Success", data.RuleArn);
    }
});
```

- For API details, see [PutRule](#) in *AWS SDK for JavaScript API Reference*.

## Send events

The following code example shows how to send Amazon EventBridge events.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { EventBridgeClient } from "@aws-sdk/client-eventbridge";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon EventBridge service client object.
export const ebClient = new EventBridgeClient({ region: REGION });
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js.
import { PutEventsCommand } from "@aws-sdk/client-eventbridge";
import { ebClient } from "./libs/eventBridgeClient.js";

// Set the parameters.
export const params = {
  Entries: [
    {
      Detail: '{ "key1": "value1", "key2": "value2" }',
      DetailType: "appRequestSubmitted",
      Resources: [
        "RESOURCE_ARN", //RESOURCE_ARN
      ],
      Source: "com.company.app",
    },
  ],
};

export const run = async () => {
  try {
    const data = await ebClient.send(new PutEventsCommand(params));
    console.log("Success, event sent; requestID:", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Uncomment this line to run execution within this file.
// run();
```

- For API details, see [PutEvents](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create CloudWatchEvents service object
var ebevents = new AWS.EventBridge({apiVersion: '2015-10-07'});

var params = {
  Entries: [
    {
      Detail: '{ \"key1\": \"value1\", \"key2\": \"value2\" }',
      DetailType: 'appRequestSubmitted',
      Resources: [
        'RESOURCE_ARN',
      ],
      Source: 'com.company.app'
    }
  ]
};

ebevents.putEvents(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Entries);
  }
});
```

- For API details, see [PutEvents](#) in *AWS SDK for JavaScript API Reference*.

## AWS Glue examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with AWS Glue.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 2997\)](#)
- [Scenarios \(p. 3004\)](#)

## Actions

### Create a crawler

The following code example shows how to create an AWS Glue crawler.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};
```

- For API details, see [CreateCrawler](#) in *AWS SDK for JavaScript API Reference*.

## Create a job definition

The following code example shows how to create an AWS Glue job definition.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};
```

- For API details, see [CreateJob](#) in *AWS SDK for JavaScript API Reference*.

## Delete a crawler

The following code example shows how to delete an AWS Glue crawler.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });

  return client.send(command);
};
```

- For API details, see [DeleteCrawler](#) in *AWS SDK for JavaScript API Reference*.

## Delete a database from the Data Catalog

The following code example shows how to delete a database from the AWS Glue Data Catalog.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const deleteDatabase = (databaseName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};
```

- For API details, see [DeleteDatabase](#) in *AWS SDK for JavaScript API Reference*.

## Delete a job definition

The following code example shows how to delete an AWS Glue job definition and all associated runs.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const deleteJob = (jobName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new DeleteJobCommand({
    JobName: jobName,
  });
```

```
    return client.send(command);
};
```

- For API details, see [DeleteJob](#) in *AWS SDK for JavaScript API Reference*.

## Delete a table from a database

The following code example shows how to delete a table from an AWS Glue Data Catalog database.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new DeleteTableCommand({
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};
```

- For API details, see [DeleteTable](#) in *AWS SDK for JavaScript API Reference*.

## Get a crawler

The following code example shows how to get an AWS Glue crawler.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getCrawler = (name) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- For API details, see [GetCrawler](#) in *AWS SDK for JavaScript API Reference*.

## Get a database from the Data Catalog

The following code example shows how to get a database from the AWS Glue Data Catalog.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getDatabase = (name) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};
```

- For API details, see [GetDatabase](#) in *AWS SDK for JavaScript API Reference*.

## Get a job run

The following code example shows how to get an AWS Glue job run.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({ region: DEFAULT_REGION });
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};
```

- For API details, see [GetJobRun](#) in *AWS SDK for JavaScript API Reference*.

## Get databases from the Data Catalog

The following code example shows how to get a list of databases from the AWS Glue Data Catalog.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getDatabases = () => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new GetDatabasesCommand({});

  return client.send(command);
};
```

- For API details, see [GetDatabases](#) in *AWS SDK for JavaScript API Reference*.

## Get job from the Data Catalog

The following code example shows how to get a job from the AWS Glue Data Catalog.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getJob = (jobName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new GetJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- For API details, see [GetJob](#) in *AWS SDK for JavaScript API Reference*.

## Get runs of a job

The following code example shows how to get runs of an AWS Glue job.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- For API details, see [GetJobRuns](#) in *AWS SDK for JavaScript API Reference*.

## Get tables from a database

The following code example shows how to get tables from a database in the AWS Glue Data Catalog.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getTables = (databaseName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};
```

- For API details, see [GetTables](#) in *AWS SDK for JavaScript API Reference*.

## List job definitions

The following code example shows how to list AWS Glue job definitions.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const listJobs = () => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new ListJobsCommand({});

  return client.send(command);
};
```

- For API details, see [ListJobs](#) in *AWS SDK for JavaScript API Reference*.

## Start a crawler

The following code example shows how to start an AWS Glue crawler.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const startCrawler = (name) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- For API details, see [StartCrawler](#) in *AWS SDK for JavaScript API Reference*.

## Start a job run

The following code example shows how to start an AWS Glue job run.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      '--input_database': dbName,
      '--input_table': tableName,
      '--output_bucket_url': `s3://${bucketName}/`
    }
  });

  return client.send(command);
};
```

- For API details, see [StartJobRun](#) in *AWS SDK for JavaScript API Reference*.

## Scenarios

### Get started running crawlers and jobs

The following code example shows how to:

- Create and run a crawler that crawls a public Amazon Simple Storage Service (Amazon S3) bucket and generates a metadata database that describes the CSV-formatted data it finds.
- List information about databases and tables in your AWS Glue Data Catalog.
- Create and run a job that extracts CSV data from the source Amazon S3 bucket, transforms it by removing and renaming fields, and loads JSON-formatted output into another Amazon S3 bucket.
- List information about job runs and view some of the transformed data.
- Delete all resources created by the demo.

For more information, see [Tutorial: Getting started with AWS Glue Studio](#).

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create and run a crawler that crawls a public Amazon Simple Storage Service (Amazon S3) bucket and generates a metadata database that describes the CSV-formatted data it finds.

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
```

```
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};

const getCrawler = (name) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const startCrawler = (name) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const crawlerExists = async ({ getCrawler }, crawlerName) => {
  try {
    await getCrawler(crawlerName);
    return true;
  } catch {
    return false;
  }
};

const makeCreateCrawlerStep = (actions) => async (context) => {
  if (await crawlerExists(actions, process.env.CRAWLER_NAME)) {
    log("Crawler already exists. Skipping creation.");
  } else {
    await actions.createCrawler(
      process.env.CRAWLER_NAME,
      process.env.ROLE_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_PREFIX,
      process.env.S3_TARGET_PATH
    );

    log("Crawler created successfully.", { type: "success" });
  }

  return { ...context };
};

const waitForCrawler = async (getCrawler, crawlerName) => {
  const waitTimeInSeconds = 30;
  const { Crawler } = await getCrawler(crawlerName);

  if (!Crawler) {
    throw new Error(`Crawler with name ${crawlerName} not found.`);
  }

  if (Crawler.State === "READY") {
```

```
        return;
    }

    log(`Crawler is ${Crawler.State}. Waiting ${waitTimeInSeconds} seconds...`);
    await wait(waitTimeInSeconds);
    return waitForCrawler(getCrawler, crawlerName);
};

const makeStartCrawlerStep =
  ({ startCrawler, getCrawler }) =>
  async (context) => {
    log("Starting crawler.");
    await startCrawler(process.env.CRAWLER_NAME);
    log("Crawler started.", { type: "success" });

    log("Waiting for crawler to finish running. This can take a while.");
    await waitForCrawler(getCrawler, process.env.CRAWLER_NAME);
    log("Crawler ready.", { type: "success" });

    return { ...context };
};
```

List information about databases and tables in your AWS Glue Data Catalog.

```
const getDatabase = (name) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};

const getTables = (databaseName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};

const makeGetDatabaseStep =
  ({ getDatabase }) =>
  async (context) => {
    const {
      Database: { Name },
    } = await getDatabase(process.env.DATABASE_NAME);
    log(`Database: ${Name}`);
    return { ...context };
};

const makeGetTablesStep =
  ({ getTables }) =>
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME);
    log("Tables:");
    log(TableList.map((table) => `  • ${table.Name}\n`));
    return { ...context };
};
```

Create and run a job that extracts CSV data from the source Amazon S3 bucket, transforms it by removing and renaming fields, and loads JSON-formatted output into another Amazon S3 bucket.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};

const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
      "--output_bucket_url": `s3://${bucketName}/`,
    }
  });

  return client.send(command);
};

const makeCreateJobStep =
  ({ createJob }) =>
  async (context) => {
    log("Creating Job.");
    await createJob(
      process.env.JOB_NAME,
      process.env.ROLE_NAME,
      process.env.BUCKET_NAME,
      process.env.PYTHON_SCRIPT_KEY
    );
    log("Job created.", { type: "success" });

    return { ...context };
};

const waitForJobRun = async (getJobRun, jobName, jobRunId) => {
  const waitTimeInSeconds = 30;
  const { JobRun } = await getJobRun(jobName, jobRunId);

  if (!JobRun) {
    throw new Error(`Job run with id ${jobRunId} not found.`);
  }

  switch (JobRun.JobRunState) {
    case "FAILED":
    case "STOPPED":
    case "TIMEOUT":
    case "STOPPED":
      throw new Error(
        `Job ${JobRun.JobRunState}. Error: ${JobRun.ErrorMessage}`
      );
  }
}
```

```

        );
    case "RUNNING":
        break;
    case "SUCCEEDED":
        return;
    default:
        throw new Error(`Unknown job run state: ${JobRun.JobRunState}`);
    }

    log(
        `Job ${JobRun.JobRunState}. Waiting ${waitTimeInSeconds} more seconds...` );
    await wait(waitTimeInSeconds);
    return waitForJobRun(getJobRun, jobName, jobRunId);
};

const promptToOpen = async (context) => {
    const { shouldOpen } = await context.prompter.prompt({
        name: "shouldOpen",
        type: "confirm",
        message: "Open the output bucket in your browser?",
    });

    if (shouldOpen) {
        return open(
            `https://s3.console.aws.amazon.com/s3/buckets/${process.env.BUCKET_NAME}?region=${DEFAULT_REGION}&tab=objects to view the output.`);
    }
};

const makeStartJobRunStep =
    ({ startJobRun, getJobRun }) =>
    async (context) => {
        log("Starting job.");
        const { JobRunId } = await startJobRun(
            process.env.JOB_NAME,
            process.env.DATABASE_NAME,
            process.env.TABLE_NAME,
            process.env.BUCKET_NAME
        );
        log("Job started.", { type: "success" });

        log("Waiting for job to finish running. This can take a while.");
        await waitForJobRun(getJobRun, process.env.JOB_NAME, JobRunId);
        log("Job run succeeded.", { type: "success" });

        await promptToOpen(context);

        return { ...context };
};

```

List information about job runs and view some of the transformed data.

```

const getJobRuns = (jobName) => {
    const client = new GlueClient({ region: DEFAULT_REGION });
    const command = new GetJobRunsCommand({
        JobName: jobName,
    });

    return client.send(command);
};

const getJobRun = (jobName, jobRunId) => {

```

```
const client = new GlueClient({ region: DEFAULT_REGION });
const command = new GetJobRunCommand({
  JobName: jobName,
  RunId: jobRunId,
});

return client.send(command);
};

const logJobRunDetails = async (getJobRun, jobName, jobRunId) => {
  const { JobRun } = await getJobRun(jobName, jobRunId);
  log(JobRun, { type: "object" });
};

const makePickJobRunStep =
  ({ getJobRuns, getJobRun }) =>
  async (context) => {
    if (context.selectedJobName) {
      const { JobRuns } = await getJobRuns(context.selectedJobName);

      const { jobRunId } = await context.prompter.prompt({
        name: "jobRunId",
        type: "list",
        message: "Select a job run to see details.",
        choices: JobRuns.map((run) => run.Id),
      });

      logJobRunDetails(getJobRun, context.selectedJobName, jobRunId);
    }

    return { ...context };
};


```

Delete all resources created by the demo.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new DeleteJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};

const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new DeleteTableCommand({
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};

const deleteDatabase = (databaseName) => {
  const client = new GlueClient({ region: DEFAULT_REGION });

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};
```

```
};

const deleteCrawler = (crawlerName) => {
    const client = new GlueClient({ region: DEFAULT_REGION });

    const command = new DeleteCrawlerCommand({
        Name: crawlerName,
    });

    return client.send(command);
};

const handleDeleteJobs = async (deleteJobFn, jobNames, context) => {
    const { selectedJobNames } = await context.prompter.prompt({
        name: "selectedJobNames",
        type: "checkbox",
        message: "Let's clean up jobs. Select jobs to delete.",
        choices: jobNames,
    });

    if (selectedJobNames.length === 0) {
        log("No jobs selected.");
    } else {
        log("Deleting jobs.");
        await Promise.all(
            selectedJobNames.map((n) => deleteJobFn(n).catch(console.error))
        );
        log("Jobs deleted.", { type: "success" });
    }
};

const makeCleanUpJobsStep =
({ listJobs, deleteJob }) =>
async (context) => {
    const { JobNames } = await listJobs();
    if (JobNames.length > 0) {
        await handleDeleteJobs(deleteJob, JobNames, context);
    }

    return { ...context };
};

const deleteTables = (deleteTable, databaseName, tableNames) =>
Promise.all(
    tableNames.map((tableName) =>
        deleteTable(databaseName, tableName).catch(console.error)
    )
);

const makeCleanUpTablesStep =
({ getTables, deleteTable }) =>
async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME).catch(
        () => ({ TableList: null })
    );

    if (TableList && TableList.length > 0) {
        const { tableNames } = await context.prompter.prompt({
            name: "tableNames",
            type: "checkbox",
            message: "Let's clean up tables. Select tables to delete.",
            choices: TableList.map((t) => t.Name),
        });

        if (tableNames.length === 0) {
            log("No tables selected.");
        } else {
            await Promise.all(
                tableNames.map((name) => deleteTable(databaseName, name).catch(console.error))
            );
            log(`Tables deleted. ${tableNames.length} tables deleted.`);
        }
    }
};
```

```
        } else {
          log("Deleting tables.");
          await deleteTables(deleteTable, process.env.DATABASE_NAME, tableNames);
          log("Tables deleted.", { type: "success" });
        }
      }

      return { ...context };
    };

const deleteDatabases = (deleteDatabase, databaseNames) =>
  Promise.all(
    databaseNames.map((dbName) => deleteDatabase(dbName).catch(console.error))
  );

const makeCleanUpDatabasesStep =
  ({ getDatabases, deleteDatabase }) =>
  async (context) => {
    const { DatabaseList } = await getDatabases();

    if (DatabaseList.length > 0) {
      const { dbNames } = await context.prompter.prompt({
        name: "dbNames",
        type: "checkbox",
        message: "Let's clean up databases. Select databases to delete.",
        choices: DatabaseList.map((db) => db.Name),
      });

      if (dbNames.length === 0) {
        log("No databases selected.");
      } else {
        log("Deleting databases.");
        await deleteDatabases(deleteDatabase, dbNames);
        log("Databases deleted.", { type: "success" });
      }
    }

    return { ...context };
  };

const cleanUpCrawlerStep = async (context) => {
  log(`Deleting crawler.`);

  try {
    await deleteCrawler(process.env.CRAWLER_NAME);
    log("Crawler deleted.", { type: "success" });
  } catch (err) {
    if (err.name === "EntityNotFoundException") {
      log(`Crawler is already deleted.`);
    } else {
      throw err;
    }
  }

  return { ...context };
};
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.
  - [CreateCrawler](#)
  - [CreateJob](#)
  - [DeleteCrawler](#)
  - [DeleteDatabase](#)

- [DeleteJob](#)
- [DeleteTable](#)
- [GetCrawler](#)
- [GetDatabase](#)
- [GetDatabases](#)
- [GetJob](#)
- [GetJobRun](#)
- [GetJobRuns](#)
- [GetTables](#)
- [ListJobs](#)
- [StartCrawler](#)
- [StartJobRun](#)

## IAM examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with AWS Identity and Access Management.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3012\)](#)
- [Scenarios \(p. 3048\)](#)

## Actions

### Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Attach the policy.

```
// Import required AWS SDK clients and commands for Node.js.
```

```
import { iamClient } from "./libs/iamClient.js";
import {
  ListAttachedRolePoliciesCommand,
  AttachRolePolicyCommand,
} from "@aws-sdk/client-iam";

// Set the parameters.
const ROLENAMESPACE = "ROLE_NAME";
const paramsRoleList = { RoleName: ROLENAMESPACE }; //ROLE_NAME
export const params = {
  PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
  RoleName: ROLENAMESPACE,
};
export const run = async () => {
  try {
    const data = await iamClient.send(
      new ListAttachedRolePoliciesCommand(paramsRoleList)
    );
    const myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (_val, index) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        console.log(
          "AmazonDynamoDBFullAccess is already attached to this role."
        );
        process.exit();
      }
    });
    try {
      const data = await iamClient.send(new AttachRolePolicyCommand(params));
      console.log("Role attached successfully");
      return data;
    } catch (err) {
      console.log("Error", err);
    }
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [AttachRolePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var paramsRoleList = {
  RoleName: process.argv[2]
};

iam.listAttachedRolePolicies(paramsRoleList, function(err, data) {
  if (err) {
```

```
        console.log("Error", err);
    } else {
        var myRolePolicies = data.AttachedPolicies;
        myRolePolicies.forEach(function (val, index, array) {
            if (myRolePolicies[index].PolicyName === 'AmazonDynamoDBFullAccess') {
                console.log("AmazonDynamoDBFullAccess is already attached to this role.")
                process.exit();
            }
        });
        var params = {
            PolicyArn: 'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess',
            RoleName: process.argv[2]
        };
        iam.attachRolePolicy(params, function(err, data) {
            if (err) {
                console.log("Unable to attach policy to role", err);
            } else {
                console.log("Role attached successfully");
            }
        });
    });
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [AttachRolePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## Create a policy

The following code example shows how to create an IAM policy.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Create the policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { CreatePolicyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const myManagedPolicy = {
    Version: "2012-10-17",
    Statement: [
        {
            Effect: "Allow",
            Action: "logs:CreateLogGroup",
            Resource: "RESOURCE_ARN", // RESOURCE_ARN
```

```
        },
        {
          Effect: "Allow",
          Action: [
            "dynamodb>DeleteItem",
            "dynamodb>GetItem",
            "dynamodb>PutItem",
            "dynamodb>Scan",
            "dynamodb>UpdateItem",
          ],
          Resource: "DYNAMODB_POLICY_NAME", // DYNAMODB_POLICY_NAME; For example,
          "myDynamoDBName".
        },
      ],
    };
  export const params = {
    PolicyDocument: JSON.stringify(myManagedPolicy),
    PolicyName: "IAM_POLICY_NAME",
  };

  export const run = async () => {
    try {
      const data = await iamClient.send(new CreatePolicyCommand(params));
      console.log("Success", data);
      return data;
    } catch (err) {
      console.log("Error", err);
    }
  };
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreatePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var myManagedPolicy = {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs>CreateLogGroup",
      "Resource": "RESOURCE_ARN"
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb>DeleteItem",
        "dynamodb>GetItem",
        "dynamodb>PutItem",
        "dynamodb>Scan",
```

```
        "dynamodb:UpdateItem"
    ],
    "Resource": "RESOURCE_ARN"
}
};

var params = {
    PolicyDocument: JSON.stringify(myManagedPolicy),
    PolicyName: 'myDynamoDBPolicy',
};

iam.createPolicy(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreatePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## Create a role

The following code example shows how to create an IAM role.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Create the role.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { CreateRoleCommand } from "@aws-sdk/client-iam";

// Sample assume role policy JSON.
const role_json = {
    Version: "2012-10-17",
    Statement: [
        {
            Effect: "Allow",
            Principal: {
                AWS: "USER_ARN", // The ARN of the user.
            },
            Action: "sts:AssumeRole",
        },
    ],
};
```

```
        ],
};

// Stringify the assume role policy JSON.
const myJson = JSON.stringify(role_json);

// Set the parameters.
const params = {
    AssumeRolePolicyDocument: myJson,
    Path: "/",
    RoleName: "ROLE_NAME"
};

const run = async () => {
    try {
        const data = await iamClient.send(new CreateRoleCommand(params));
        console.log("Success. Role created. Role Arn: ", data.Role.RoleName);
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For API details, see [CreateRole](#) in *AWS SDK for JavaScript API Reference*.

## Create a service-linked role

The following code example shows how to create an IAM service-linked role.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Create a service-linked role.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { CreateServiceLinkedRoleCommand } from "@aws-sdk/client-iam";
// Set the parameters.
const params = {
    AWSServiceName: "AWS_SERVICE_NAME" /* required */,
};

const run = async () => {
    try {
        const data = await iamClient.send(
            new CreateServiceLinkedRoleCommand(params)
        );
        console.log("Success", data);
    } catch (err) {
```

```
        console.log("Error", err);
    }
};

run();
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for JavaScript API Reference*.

## Create a user

The following code example shows how to create an IAM user.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Create the user.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { GetUserCommand, CreateUserCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { UserName: "USER_NAME" }; //USER_NAME

export const run = async () => {
    try {
        const data = await iamClient.send(new GetUserCommand(params));
        console.log(
            "User " + "USER_NAME" + " already exists",
            data.UserId
        );
        return data;
    } catch (err) {
        try {
            const results = await iamClient.send(new CreateUserCommand(params));
            console.log("Success", results);
            return results;
        } catch (err) {
            console.log("Error", err);
        }
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateUser](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  UserName: process.argv[2]
};

iam.getUser(params, function(err, data) {
  if (err && err.code === 'NoSuchEntity') {
    iam.createUser(params, function(err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  } else {
    console.log("User " + process.argv[2] + " already exists", data.User.UserId);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateUser](#) in [AWS SDK for JavaScript API Reference](#).

## Create an access key

The following code example shows how to create an IAM access key.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Create the access key.

```
// Import required AWS SDK clients and commands for Node.js.
```

```
import { iamClient } from "./libs/iamClient.js";
import { CreateAccessKeyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {UserName: "IAM_USER_NAME"}; //IAM_USER_NAME

export const run = async () => {
  try {
    const data = await iamClient.send(new CreateAccessKeyCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateAccessKey in AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.createAccessKey({UserName: 'IAM_USER_NAME'}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.AccessKey);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateAccessKey in AWS SDK for JavaScript API Reference](#).

### Create an alias for an account

The following code example shows how to create an alias for an IAM account.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
```

```
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

Create the account alias.

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "./libs/iamClient.js";  
import { CreateAccountAliasCommand } from "@aws-sdk/client-iam";  
  
// Set the parameters.  
export const params = { AccountAlias: "ACCOUNT_ALIAS" }; //ACCOUNT_ALIAS  
  
export const run = async () => {  
    try {  
        const data = await iamClient.send(new CreateAccountAliasCommand(params));  
        console.log("Success", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateAccountAlias](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create the IAM service object  
var iam = new AWS.IAM({apiVersion: '2010-05-08'});  
  
iam.createAccountAlias({AccountAlias: process.argv[2]}, function(err, data) {  
    if (err) {  
        console.log("Error", err);  
    } else {  
        console.log("Success", data);  
    }  
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateAccountAlias](#) in [AWS SDK for JavaScript API Reference](#).

## Delete a policy

The following code example shows how to delete an IAM policy.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Delete the policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { DeletePolicyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const params = { PolicyArn: "POLICY_ARN" };

const run = async () => {
  try {
    const data = await iamClient.send(new DeletePolicyCommand(params));
    console.log("Success. Policy deleted.", data);

  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [DeletePolicy](#) in *AWS SDK for JavaScript API Reference*.

## Delete a role

The following code example shows how to delete an IAM role.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Delete the role.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { DeleteRoleCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const params = {
    RoleName: "ROLE_NAME"
}

const run = async () => {
    try {
        const data = await iamClient.send(new DeleteRoleCommand(params));
        console.log("Success. Role deleted.", data);
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For API details, see [DeleteRole](#) in *AWS SDK for JavaScript API Reference*.

## Delete a server certificate

The following code example shows how to delete an IAM server certificate.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Delete a server certificate.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { DeleteServerCertificateCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { ServerCertificateName: "CERTIFICATE_NAME" }; // CERTIFICATE_NAME

export const run = async () => {
    try {
        const data = await iamClient.send(
            new DeleteServerCertificateCommand(params)
        );
        console.log("Success", data);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
```

```
    }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteServerCertificate](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.deleteServerCertificate({ServerCertificateName: 'CERTIFICATE_NAME'}, function(err,
  data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteServerCertificate](#) in [AWS SDK for JavaScript API Reference](#).

## Delete a user

The following code example shows how to delete an IAM user.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Delete the user.

```
// Import required AWS SDK clients and commands for Node.js.
```

```
import { iamClient } from "./libs/iamClient.js";
import { DeleteUserCommand, GetUserCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { UserName: "USER_NAME" }; //USER_NAME

export const run = async () => {
  try {
    const data = await iamClient.send(new GetUserCommand(params));
    return data;
  } catch (err) {
    console.log("Error", err);
  }
}

try {
  const results = await iamClient.send(new DeleteUserCommand(params));
  console.log("Success", results);
  return results;
} catch (err) {
  console.log("Error", err);
}

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteUser](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  UserName: process.argv[2]
};

iam.getUser(params, function(err, data) {
  if (err && err.code === 'NoSuchEntity') {
    console.log("User " + process.argv[2] + " does not exist.");
  } else {
    iam.deleteUser(params, function(err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteUser](#) in [AWS SDK for JavaScript API Reference](#).

## Delete an access key

The following code example shows how to delete an IAM access key.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Delete the access key.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { DeleteAccessKeyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  AccessKeyId: "ACCESS_KEY_ID", // ACCESS_KEY_ID
  UserName: "USER_NAME", // USER_NAME
};

export const run = async () => {
  try {
    const data = await iamClient.send(new DeleteAccessKeyCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAccessKey](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});
```

```
var params = {
  AccessKeyId: 'ACCESS_KEY_ID',
  UserName: 'USER_NAME'
};

iam.deleteAccessKey(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAccessKey](#) in [AWS SDK for JavaScript API Reference](#).

## Delete an account alias

The following code example shows how to delete an IAM account alias.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Delete the account alias.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { DeleteAccountAliasCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { AccountAlias: "ALIAS" }; // ALIAS

export const run = async () => {
  try {
    const data = await iamClient.send(new DeleteAccountAliasCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAccountAlias](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.deleteAccountAlias({AccountAlias: process.argv[2]}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteAccountAlias](#) in *AWS SDK for JavaScript API Reference*.

## Detach a policy from a role

The following code example shows how to detach an IAM policy from a role.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Detach the policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import {
  ListAttachedRolePoliciesCommand,
  DetachRolePolicyCommand,
} from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { RoleName: "ROLE_NAME" }; //ROLE_NAME

export const run = async () => {
  try {
```

```
const data = await iamClient.send(
  new ListAttachedRolePoliciesCommand(params)
);
const myRolePolicies = data.AttachedPolicies;
myRolePolicies.forEach(function (_val, index) {
  if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
    try {
      await iamClient.send(
        new DetachRolePolicyCommand(paramsRoleList)
      );
      console.log("Policy detached from role successfully");
      process.exit();
    } catch (err) {
      console.log("Unable to detach policy from role", err);
    }
  } else {
  });
  return data;
} catch (err) {
  console.log("User " + "USER_NAME" + " does not exist.");
}
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DetachRolePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var paramsRoleList = {
  RoleName: process.argv[2]
};

iam.listAttachedRolePolicies(paramsRoleList, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === 'AmazonDynamoDBFullAccess') {
        var params = {
          PolicyArn: 'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess',
          RoleName: process.argv[2]
        };
        iam.detachRolePolicy(params, function(err, data) {
          if (err) {
            console.log("Unable to detach policy from role", err);
          } else {
            console.log("Policy detached from role successfully");
            process.exit();
          }
        });
      }
    });
  }
});
```

```
        }
    );
}
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DetachRolePolicy](#) in [AWS SDK for JavaScript API Reference](#).

## Get a policy

The following code example shows how to get an IAM policy.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Get the policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { GetPolicyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const params = {
  PolicyArn: "POLICY_ARN" /* required */,
};

const run = async () => {
  try {
    const data = await iamClient.send(new GetPolicyCommand(params));
    console.log("Success", data.Policy);
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetPolicy](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
    PolicyArn: 'arn:aws:iam::aws:policy/AWSLambdaExecute'
};

iam.getPolicy(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data.Policy.Description);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetPolicy](#) in [AWS SDK for JavaScript API Reference](#).

## Get a role

The following code example shows how to get an IAM role.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Get the role.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { GetRoleCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const params = {
    RoleName: "ROLE_NAME" /* required */
};

const run = async () => {
    try {
        const data = await iamClient.send(new GetRoleCommand(params));
        console.log("Success", data.Role);
    } catch (err) {
```

```
        console.log("Error", err);
    }
};

run();
```

- For API details, see [GetRole](#) in *AWS SDK for JavaScript API Reference*.

## Get a server certificate

The following code example shows how to get an IAM server certificate.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Get a server certificate.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { GetServerCertificateCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { ServerCertificateName: "CERTIFICATE_NAME" }; //CERTIFICATE_NAME

export const run = async () => {
    try {
        const data = await iamClient.send(new GetServerCertificateCommand(params));
        console.log("Success", data);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetServerCertificate](#) in *AWS SDK for JavaScript API Reference*.

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
```

```
AWS.config.update({region: 'REGION');

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.getServerCertificate({ServerCertificateName: 'CERTIFICATE_NAME'}, function(err,
data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetServerCertificate](#) in [AWS SDK for JavaScript API Reference](#).

## Get data about the last use of an access key

The following code example shows how to get data about the last use of an IAM access key.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Get the access key.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { GetAccessKeyLastUsedCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { AccessKeyId: "ACCESS_KEY_ID" }; //ACCESS_KEY_ID

export const run = async () => {
  try {
    const data = await iamClient.send(new GetAccessKeyLastUsedCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetAccessKeyLastUsed](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.getAccessKeyLastUsed({AccessKeyId: 'ACCESS_KEY_ID'}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.AccessKeyLastUsed);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetAccessKeyLastUsed](#) in [AWS SDK for JavaScript API Reference](#).

## Get the account password policy

The following code example shows how to get the IAM account password policy.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Get the account password policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { GetAccountPasswordPolicyCommand } from "@aws-sdk/client-iam";

const run = async () => {
  try {
    const data = await iamClient.send(new GetAccountPasswordPolicyCommand({}));
    console.log("Success", data.PasswordPolicy);
  } catch (err) {
    console.log("Error", err);
  }
};
```

```
run();
```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for JavaScript API Reference*.

## List SAML providers

The following code example shows how to list SAML providers for IAM.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the SAML providers.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { ListSAMLProvidersCommand } from "@aws-sdk/client-iam";

export const run = async () => {
    try {
        const results = await iamClient.send(new ListSAMLProvidersCommand([]));
        console.log("Success", results);
        return results;
    } catch (err) {
        console.log("Error", err);
    }
}
run();
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for JavaScript API Reference*.

## List a user's access keys

The following code example shows how to list a user's IAM access keys.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
```

```
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

List the access keys.

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "./libs/iamClient.js";  
import { ListAccessKeysCommand } from "@aws-sdk/client-iam";  
  
// Set the parameters.  
export const params = {  
    MaxItems: 5,  
    UserName: "IAM_USER_NAME", //IAM_USER_NAME  
};  
  
export const run = async () => {  
    try {  
        const data = await iamClient.send(new ListAccessKeysCommand(params));  
        console.log("Success", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListAccessKeys](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create the IAM service object  
var iam = new AWS.IAM({apiVersion: '2010-05-08'});  
  
var params = {  
    MaxItems: 5,  
    UserName: 'IAM_USER_NAME'  
};  
  
iam.listAccessKeys(params, function(err, data) {  
    if (err) {  
        console.log("Error", err);  
    } else {  
        console.log("Success", data);  
    }  
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [ListAccessKeys](#) in *AWS SDK for JavaScript API Reference*.

## List account aliases

The following code example shows how to list IAM account aliases.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the account aliases.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { ListAccountAliasesCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { MaxItems: 5 };

export const run = async () => {
  try {
    const data = await iamClient.send(new ListAccountAliasesCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListAccountAliases](#) in *AWS SDK for JavaScript API Reference*.

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.listAccountAliases({MaxItems: 10}, function(err, data) {
```

```
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListAccountAliases](#) in [AWS SDK for JavaScript API Reference](#).

## List groups

The following code example shows how to list IAM groups.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the groups.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import {ListGroupsCommand} from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  RoleName: 'ROLE_NAME', /* This is a number value. Required */
  Marker: 'MARKER', /* This is a string value. Optional */
  MaxItems: 'MAX_ITEMS' /* This is a number value. Optional */
};

export const run = async () => {
  try {
    const data = await iamClient.send(new ListGroupsCommand({}));
    console.log("Success", data.Groups);
  } catch (err) {
    console.log("Error", err);
  }
}
run();
```

- For API details, see [ListGroups](#) in [AWS SDK for JavaScript API Reference](#).

## List inline policies for a role

The following code example shows how to list inline policies for an IAM role.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the policies.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import {ListRolePoliciesCommand} from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
    RoleName: 'ROLE_NAME', /* This is a number value. Required */
    Marker: 'MARKER', /* This is a string value. Optional */
    MaxItems: 'MAX_ITEMS' /* This is a number value. Optional */
};

export const run = async () => {
    try {
        const results = await iamClient.send(new ListRolePoliciesCommand(params));
        console.log("Success", results);
        return results;
    } catch (err) {
        console.log("Error", err);
    }
}
run();
```

- For API details, see [ListRolePolicies in AWS SDK for JavaScript API Reference](#).

## List policies

The following code example shows how to list IAM policies.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
```

```
export { iamClient };
```

List the policies.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import {ListPoliciesCommand} from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
    Marker: 'MARKER',
    MaxItems: 'MAX_ITEMS',
    OnlyAttached: "ONLY_ATTACHED", /* Options are "true" or "false"*/
    PathPrefix: 'PATH_PREFIX',
    PolicyUsageFilter: "POLICY_USAGE_FILTER", /* Options are "PermissionsPolicy" or
"PermissionsBoundary"*/
    Scope: "SCOPE" /* Options are "All", "AWS", "Local"*/
};

export const run = async () => {
    try {
        const results = await iamClient.send(new ListPoliciesCommand(params));
        console.log("Success", results);
        return results;
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For API details, see [ListPolicies](#) in *AWS SDK for JavaScript API Reference*.

## List policies attached to a role

The following code example shows how to list policies attached to an IAM role.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the policies that are attached to a role.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import {ListAttachedRolePoliciesCommand} from "@aws-sdk/client-iam";
```

```
// Set the parameters.
export const params = {
  RoleName: 'ROLE_NAME' /* required */
};

export const run = async () => {
  try {
    const data = await iamClient.send(new ListAttachedRolePoliciesCommand(params));
    console.log("Success", data.AttachedPolicies);
  } catch (err) {
    console.log("Error", err);
  }
}
run();
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for JavaScript API Reference*.

## List roles

The following code example shows how to list IAM roles.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the roles.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { ListRolesCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const params = {
  Marker: 'MARKER', // This is a string value.
  MaxItems: 'MAX_ITEMS' // This is a number value.
};

const run = async () => {
  try {
    const results = await iamClient.send(new ListRolesCommand(params));
    console.log("Success", results);
    return results;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [ListRoles](#) in *AWS SDK for JavaScript API Reference*.

## List server certificates

The following code example shows how to list IAM server certificates.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the certificates.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { ListServerCertificatesCommand } from "@aws-sdk/client-iam";

export const run = async () => {
  try {
    const data = await iamClient.send(new ListServerCertificatesCommand({}));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListServerCertificates](#) in *AWS SDK for JavaScript API Reference*.

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.listServerCertificates({}, function(err, data) {
  if (err) {
    console.log("Error", err);
```

```
    } else {
      console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListServerCertificates](#) in [AWS SDK for JavaScript API Reference](#).

## List users

The following code example shows how to list IAM users.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

List the users.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { ListUsersCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { MaxItems: 10 };

export const run = async () => {
  try {
    const data = await iamClient.send(new ListUsersCommand(params));
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListUsers](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  MaxItems: 10
};

iam.listUsers(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var users = data.Users || [];
    users.forEach(function(user) {
      console.log("User " + user.UserName + " created", user.CreateDate);
    });
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListUsers](#) in [AWS SDK for JavaScript API Reference](#).

## Update a server certificate

The following code example shows how to update an IAM server certificate.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Update a server certificate.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { UpdateServerCertificateCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  ServerCertificateName: "CERTIFICATE_NAME", //CERTIFICATE_NAME
  NewServerCertificateName: "NEW_CERTIFICATE_NAME", //NEW_CERTIFICATE_NAME
};

export const run = async () => {
```

```
try {
  const data = await iamClient.send(
    new UpdateServerCertificateCommand(params)
  );
  console.log("Success", data);
  return data;
} catch (err) {
  console.log("Error", err);
}
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateServerCertificate](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  ServerCertificateName: 'CERTIFICATE_NAME',
  NewServerCertificateName: 'NEW_CERTIFICATE_NAME'
};

iam.updateServerCertificate(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateServerCertificate](#) in [AWS SDK for JavaScript API Reference](#).

## Update a user

The following code example shows how to update an IAM user.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
```

```
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

Update the user.

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "./libs/iamClient.js";  
import { UpdateUserCommand } from "@aws-sdk/client-iam";  
  
// Set the parameters.  
export const params = {  
    UserName: "ORIGINAL_USER_NAME", //ORIGINAL_USER_NAME  
    NewUserName: "NEW_USER_NAME", //NEW_USER_NAME  
};  
  
export const run = async () => {  
    try {  
        const data = await iamClient.send(new UpdateUserCommand(params));  
        console.log("Success, username updated");  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateUser](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js  
var AWS = require('aws-sdk');  
// Set the region  
AWS.config.update({region: 'REGION'});  
  
// Create the IAM service object  
var iam = new AWS.IAM({apiVersion: '2010-05-08'});  
  
var params = {  
    UserName: process.argv[2],  
    NewUserName: process.argv[3]  
};  
  
iam.updateUser(params, function(err, data) {  
    if (err) {  
        console.log("Error", err);  
    } else {  
        console.log("Success", data);  
    }  
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateUser](#) in [AWS SDK for JavaScript API Reference](#).

## Update an access key

The following code example shows how to update an IAM access key.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

Update the access key.

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "./libs/iamClient.js";
import { UpdateAccessKeyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  AccessKeyId: "ACCESS_KEY_ID", //ACCESS_KEY_ID
  Status: "Active",
  UserName: "USER_NAME", //USER_NAME
};

export const run = async () => {
  try {
    const data = await iamClient.send(new UpdateAccessKeyCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateAccessKey](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
```

```
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
    AccessKeyId: 'ACCESS_KEY_ID',
    Status: 'Active',
    UserName: 'USER_NAME'
};

iam.updateAccessKey(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UpdateAccessKey](#) in [AWS SDK for JavaScript API Reference](#).

## Scenarios

### Create a user and assume a role

The following code example shows how to:

- Create a user who has no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list Amazon S3 buckets using temporary credentials.
- Delete the policy, role, and user.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
export const REGION = "REGION"; // For example, "us-east-1".
// Create an Amazon S3 service client object.
export const iamClient = new IAMClient({ region: REGION });
```

Create an IAM user and a role that grants permission to list Amazon S3 buckets. The user has rights only to assume the role. After assuming the role, use temporary credentials to list buckets for the account.

```
// Import required AWS SDK clients and commands for Node.js.
```

```
import { iamClient, REGION } from "../libs/iamClient.js"; // Helper function that
  creates an IAM service client module.
import {
  CreateUserCommand,
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  AttachUserPolicyCommand,
  DeleteAccessKeyCommand,
  DeleteUserCommand,
  DeleteRoleCommand,
  DeletePolicyCommand,
  DetachUserPolicyCommand,
  DetachRolePolicyCommand,
} from "@aws-sdk/client-iam";
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";

if (process.argv.length < 6) {
  console.log(
    "Usage: node iam_basics.js <user name> <s3 policy name> <role name> <assume
  policy name>\n" +
    "Example: node iam_basics.js test-user my-s3-policy my-iam-role my-assume-role"
  );
}
// Set the parameters.
const region = REGION;
const userName = process.argv[2];
const s3_policy_name = process.argv[3];
const role_name = process.argv[4];
const assume_policy_name = process.argv[5];

// Helper function to delay running the code while the AWS service calls wait for
// responses.
function wait(ms) {
  var start = Date.now();
  var end = start
  while (end < start + ms){
    end = Date.now()
  }
}

export const run = async (
  userName,
  s3_policy_name,
  role_name,
  assume_policy_name
) => {
  try {
    // Create a new user.
    const user_params = { UserName: userName };
    console.log("\nCreating a user name " + user_params.UserName + "...\\n");
    const data = await iamClient.send(
      new CreateUserCommand({ UserName: userName })
    );
    const user_arn = data.User.Arn;
    const user_name = data.User.UserName;
    console.log(
      "User with name" + user_name + " and ARN " + user_arn + " created."
    );
    try {
      // Create access keys for the new user.
      console.log(
        "\nCreating access keys for " + user_params.UserName + "...\\n"
      );
    }
  }
}
```

```
const access_key_params = { UserName: user_name };
const data = await iamClient.send(
    new CreateAccessKeyCommand(access_key_params)
);
console.log("Success. Access key created: ", data.AccessKey.AccessKeyId);
var myAccessKey = data.AccessKey.AccessKeyId;
var mySecretAccessKey = data.AccessKey.SecretAccessKey;

try {
    // Attempt to list S3 buckets.
    console.log(
        "\nWaiting 10 seconds for user and access keys to be created...\n"
    );
    wait(10000);
    console.log(
        "Attempt to list S3 buckets with the new user (without permissions)...\\n"
    );
    // Use the credentials for the new user that you created.
    var user_creds = {
        accessKeyId: myAccessKey,
        secretAccessKey: mySecretAccessKey,
    };
    const s3Client = new S3Client({
        credentials: user_creds,
        region: region,
    });
    await s3Client.send(new ListBucketsCommand({}));
} catch (err) {
    console.log(
        "Error. As expected the new user has no permissions to list buckets. ",
        err.stack
    );
    console.log(
        "\nCreating policy to allow the new user to list all buckets, and to assume
an STS role...\\n"
    );
    const myManagedPolicy = {
        Version: "2012-10-17",
        Statement: [
            {
                Effect: "Allow",
                Action: ["s3>ListAllMyBuckets", "sts:AssumeRole"],
                Resource: "*",
            },
        ],
    };
    const policy_params = {
        PolicyDocument: JSON.stringify(myManagedPolicy),
        PolicyName: s3_policy_name, // Name of the new policy.
    };
    const data = await iamClient.send(
        new CreatePolicyCommand(policy_params)
    );
    console.log(
        "Success. Policy created that allows listing of all S3 buckets.\\n" +
        "Policy ARN: " +
        data.Policy.Arn +
        "\\n" +
        "Policy name: " +
        data.Policy.PolicyName +
        "\\n"
    );
    var s3_policy_arn = data.Policy.Arn;

    try {
```

```
        console.log(
            "\nCreating a role with a trust policy that lets the user assume the
role....\n"
        );

        const role_json = {
            Version: "2012-10-17",
            Statement: [
                {
                    Effect: "Allow",
                    Principal: {
                        AWS: user_arn, // The ARN of the user.
                    },
                    Action: "sts:AssumeRole",
                },
            ],
        };
        const myJson = JSON.stringify(role_json);

        const role_params = {
            AssumeRolePolicyDocument: myJson, // Trust relationship policy document.
            Path: "/",
            RoleName: role_name // The name of the new role.
        };
        const data = await iamClient.send(new CreateRoleCommand(role_params));
        console.log("Success. Role created. Role Arn: ", data.Role.Arn);
        const role_arn = data.Role.Arn;
        try {
            try {
                console.log(
                    "\nAttaching to the role the policy with permissions to list all
buckets....\n"
                );
                const params = {
                    PolicyArn: s3_policy_arn,
                    RoleName: role_name,
                };
                await iamClient.send(new AttachRolePolicyCommand(params));
                console.log("Success. Policy attached successfully to role.");
            } catch {
                console.log(
                    "\nCreate a policy that enables the user to assume the role ....\n"
                );
                const myNewPolicy = {
                    Version: "2012-10-17",
                    Statement: [
                        {
                            Effect: "Allow",
                            Action: ["sts:AssumeRole"],
                            Resource: role_arn,
                        },
                    ],
                };
                const policy_params = {
                    PolicyDocument: JSON.stringify(myNewPolicy),
                    PolicyName: assume_policy_name,
                };
                const data = await iamClient.send(
                    new CreatePolicyCommand(policy_params)
                );
                console.log(
                    "Success. Policy created. Policy ARN: " + data.Policy.Arn
                );
            }
            const assume_policy_arn = data.Policy.Arn;
            try {
                console.log("\nAttaching the policy to the user....\n");
            }
        }
    }
}
```

```
const attach_policy_to_user_params = {
    PolicyArn: assume_policy_arn,
    UserName: user_name,
};
await iamClient.send(
    new AttachUserPolicyCommand(attach_policy_to_user_params)
);
console.log(
    "\nWaiting 10 seconds for policy to be attached...\n"
);
wait(10000);
console.log(
    "Success. Policy attached to user " + user_name + "."
);
try {
    console.log(
        "\nAssume for the user the role with permission to list all
buckets....\n"
    );
    const assume_role_params = {
        RoleArn: role_arn, //ARN_OF_ROLE_TO_ASSUME
        RoleSessionName: "session1",
        DurationSeconds: 900,
    };
    // Create an AWS STS client with the credentials for the user.
    Remember, the user has permissions to assume roles using AWS STS.
    const stsClientWithUsersCreds = new STSClient({
        credentials: user_creds,
        region: REGION,
    });

    const data = await stsClientWithUsersCreds.send(
        new AssumeRoleCommand(assume_role_params)
    );
    console.log(
        "Success assuming role. Access key id is " +
        data.Credentials.AccessKeyId +
        "\n" +
        "Secret access key is " +
        data.Credentials.SecretAccessKey
    );
}

const newAccessKey = data.Credentials.AccessKeyId;
const newSecretAccessKey = data.Credentials.SecretAccessKey;

console.log(
    "\nWaiting 10 seconds for the user to assume the role with
permission to list all buckets...\n"
);
wait(10000);
// Set the parameters for the temporary credentials. This grants
permission to list S3 buckets.
var new_role_creds = {
    accessKeyId: newAccessKey,
    secretAccessKey: newSecretAccessKey,
    sessionToken: data.Credentials.SessionToken,
};
try {
    console.log(
        "Listing the S3 buckets using the credentials of the assumed
role... \n"
    );
    // Create an S3 client with the temporary credentials.
    const s3ClientWithNewCreds = new S3Client({
        credentials: new_role_creds,
        region: REGION,
```

```
});
const data = await s3ClientWithNewCreds.send(
    new ListBucketsCommand({})
);
console.log("Success. Your S3 buckets are:", data.Buckets);
try {
    console.log(
        "Detaching s3 policy from user " + userName + " ... \n"
    );
    await iamClient.send(
        new DetachUserPolicyCommand({
            PolicyArn: assume_policy_arn,
            UserName: userName,
        })
    );
    console.log("Success, S3 policy detached from user.");
    try {
        console.log(
            "Detaching role policy from " + role_name + " ... \n"
        );
        await iamClient.send(
            new DetachRolePolicyCommand({
                PolicyArn: s3_policy_arn,
                RoleName: role_name,
            })
        );
        console.log(
            "Success, assume policy detached from role."
        );
        try {
            console.log("Deleting s3 policy ... \n");
            await iamClient.send(
                new DeletePolicyCommand({
                    PolicyArn: s3_policy_arn,
                })
            );
            console.log("Success, S3 policy deleted.");
            try {
                console.log("Deleting assume role policy ... \n");
                await iamClient.send(
                    new DeletePolicyCommand({
                        PolicyArn: assume_policy_arn,
                    })
                );
                try {
                    console.log("Deleting access keys ... \n");
                    await iamClient.send(
                        new DeleteAccessKeyCommand({
                            UserName: userName,
                            AccessKeyId: myAccessKey,
                        })
                    );
                    try {
                        console.log(
                            "Deleting user " + user_name + " ... \n"
                        );
                        await iamClient.send(
                            new DeleteUserCommand({ UserName: userName })
                        );
                        console.log("Success, user deleted.");
                        try {
                            console.log(
                                "Deleting role " + role_name + " ... \n"
                            );
                            await iamClient.send(
                                new DeleteRoleCommand({

```

```
        RoleName: role_name,
    })
);
console.log("Success, role deleted.");
return "Run successfully"; // For unit tests.
} catch (err) {
    console.log("Error deleting role .", err);
}
} catch (err) {
    console.log("Error deleting user.", err);
}
} catch (err) {
    console.log("Error deleting access keys.", err);
}
} catch (err) {
    console.log(
        "Error detaching assume role policy from user.",
        err
    );
}
} catch (err) {
    console.log("Error deleting role.", err);
}
} catch (err) {
    console.log("Error deleting user.", err);
}
} catch (err) {
    console.log("Error detaching S3 policy from role.", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error listing S3 buckets.", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error assuming role.", err);
    process.exit(1);
}
} catch (err) {
    console.log(
        "Error adding permissions to user to assume role.",
        err
    );
    process.exit(1);
}
} catch (err) {
    console.log("Error assuming role.", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error creating policy. ", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error attaching policy to role.", err);
    process.exit(1);
}
}
} catch (err) {
    console.log("Error creating access keys. ", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error creating user. ", err);
}
};
```

```
run(userName, s3_policy_name, role_name, assume_policy_name);
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## Lambda examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with AWS Lambda.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3055\)](#)
- [Scenarios \(p. 3058\)](#)

## Actions

### Create a function

The following code example shows how to create a Lambda function.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const createFunction = async (funcName, roleArn) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const code = await zip(`.${dirname}../functions/${funcName}`);

  const command = new CreateFunctionCommand({
    Code: { ZipFile: code },
    FunctionName: funcName,
    Role: roleArn,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
```

```
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};
```

- For API details, see [CreateFunction](#) in *AWS SDK for JavaScript API Reference*.

## Delete a function

The following code example shows how to delete a Lambda function.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const deleteFunction = (funcName) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const command = new DeleteFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- For API details, see [DeleteFunction](#) in *AWS SDK for JavaScript API Reference*.

## Get a function

The following code example shows how to get a Lambda function.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const getFunction = (funcName) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const command = new GetFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- For API details, see [GetFunction](#) in *AWS SDK for JavaScript API Reference*.

## Invoke a function

The following code example shows how to invoke a Lambda function.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const invoke = async (funcName, payload) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const command = new InvokeCommand({
    FunctionName: funcName,
    Payload: JSON.stringify(payload),
    LogType: LogType.Tail,
  });

  const { Payload, LogResult } = await client.send(command);
  const result = Buffer.from(Payload).toString();
  const logs = Buffer.from(LogResult, "base64").toString();
  return { logs, result };
};
```

- For API details, see [Invoke](#) in *AWS SDK for JavaScript API Reference*.

## List functions

The following code example shows how to list Lambda functions.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const listFunctions = async () => {
  const client = createClientForDefaultRegion(LambdaClient);
  const command = new ListFunctionsCommand({});

  return client.send(command);
};
```

- For API details, see [ListFunctions](#) in *AWS SDK for JavaScript API Reference*.

## Update function code

The following code example shows how to update Lambda function code.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const updateFunctionCode = async (funcName, newFunc) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const code = await zip(`.${dirname}..functions/${newFunc}`);
  const command = new UpdateFunctionCodeCommand({
    ZipFile: code,
    FunctionName: funcName,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });
};
```

```
    });
    return client.send(command);
};
```

- For API details, see [UpdateFunctionCode](#) in *AWS SDK for JavaScript API Reference*.

## Update function configuration

The following code example shows how to update Lambda function configuration.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
const updateFunctionConfiguration = async (funcName) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const config = readFileSync(
    `${dirname}../functions/${funcName}/config.json`
  ).toString();
  const command = new UpdateFunctionConfigurationCommand({
    ...JSON.parse(config),
    FunctionName: funcName,
  });
  return client.send(command);
};
```

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for JavaScript API Reference*.

## Scenarios

### Get started with functions

The following code example shows how to:

- Create an AWS Identity and Access Management (IAM) role that grants Lambda permission to write to logs.
- Create a Lambda function and upload handler code.
- Invoke the function with a single parameter and get results.
- Update the function code and configure its Lambda environment with an environment variable.
- Invoke the function with new parameters and get results. Display the execution log that's returned from the invocation.
- List the functions for your account.
- Delete the IAM role and the Lambda function.

For more information, see [Create a Lambda function with the console](#).

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an AWS Identity and Access Management (IAM) role that grants Lambda permission to write to logs.

```
log(`Creating role (${NAME_ROLE_LAMBDA})...`);  
const response = await createRole({  
    AssumeRolePolicyDocument: parseString({  
        Version: "2012-10-17",  
        Statement: [  
            {  
                Effect: "Allow",  
                Principal: {  
                    Service: "lambda.amazonaws.com",  
                },  
                Action: "sts:AssumeRole",  
            },  
        ],  
    }),  
    RoleName: NAME_ROLE_LAMBDA,  
});  
  
const attachRolePolicy = async (roleName, policyArn) => {  
    const client = createClientForDefaultRegion(IAMClient);  
    const command = new AttachRolePolicyCommand({  
        PolicyArn: policyArn, // For example, arn:aws:iam::aws:policy/service-role/  
        AWSLambdaBasicExecutionRole  
        RoleName: roleName, // For example, lambda-basic-execution-role  
    });  
  
    return client.send(command);  
};
```

Create a Lambda function and upload handler code.

```
const createFunction = async (funcName, roleArn) => {  
    const client = createClientForDefaultRegion(LambdaClient);  
    const code = await zip(`.${dirname}../functions/${funcName}`);  
  
    const command = new CreateFunctionCommand({  
        Code: { ZipFile: code },  
        FunctionName: funcName,  
        Role: roleArn,  
        Architectures: [Architecture.arm64],  
        Handler: "index.handler", // Required when sending a .zip file  
        PackageType: PackageType.Zip, // Required when sending a .zip file  
        Runtime: Runtime.nodejs16x, // Required when sending a .zip file  
    });  
  
    return client.send(command);  
};
```

Invoke the function with a single parameter and get results.

```
const invoke = async (funcName, payload) => {  
    const client = createClientForDefaultRegion(LambdaClient);  
    const command = new InvokeCommand({  
        FunctionName: funcName,  
        Payload: JSON.stringify(payload),  
        LogType: LogType.Tail,  
    });
```

```
const { Payload, LogResult } = await client.send(command);
const result = Buffer.from(Payload).toString();
const logs = Buffer.from(LogResult, "base64").toString();
return { logs, result };
};
```

Update the function code and configure its Lambda environment with an environment variable.

```
const updateFunctionCode = async (funcName, newFunc) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const code = await zip(`.${dirname}../functions/${newFunc}`);
  const command = new UpdateFunctionCodeCommand({
    ZipFile: code,
    FunctionName: funcName,
    Architectures: [Architecture.arm64],
    Handler: "index.handler", // Required when sending a .zip file
    PackageType: PackageType.Zip, // Required when sending a .zip file
    Runtime: Runtime.nodejs16x, // Required when sending a .zip file
  });

  return client.send(command);
};

const updateFunctionConfiguration = async (funcName) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const config = readFileSync(
    `.${dirname}../functions/${funcName}/config.json`
  ).toString();
  const command = new UpdateFunctionConfigurationCommand({
    ...JSON.parse(config),
    FunctionName: funcName,
  });
  return client.send(command);
};
```

List the functions for your account.

```
const listFunctions = async () => {
  const client = createClientForDefaultRegion(LambdaClient);
  const command = new ListFunctionsCommand({});

  return client.send(command);
};
```

Delete the IAM role and the Lambda function.

```
const deleteRole = (roleName) => {
  const client = createClientForDefaultRegion(IAMClient);
  const command = new DeleteRoleCommand({ RoleName: roleName });
  return client.send(command);
};

const deleteFunction = (funcName) => {
  const client = createClientForDefaultRegion(LambdaClient);
  const command = new DeleteFunctionCommand({ FunctionName: funcName });
  return client.send(command);
};
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## Amazon Personalize examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon Personalize.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3061\)](#)

## Actions

### Create a batch interface job

The following code example shows how to create a Amazon Personalize batch interface job.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchInferenceJobCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch inference job's parameters.

export const createBatchInferenceJobParam = {
  jobName: 'JOB_NAME',
  jobInput: { /* required */
    s3DataSource: { /* required */
      path: 'INPUT_PATH', /* required */
      // kmsKeyArn: 'INPUT_KMS_KEY_ARN' /* optional */
    }
  },
  jobOutput: { /* required */
    s3DataDestination: { /* required */
      path: 'OUTPUT_PATH', /* required */
    }
  }
};
```

```
// kmsKeyArn: 'OUTPUT_KMS_KEY_ARN' /* optional */
    }
},
roleArn: 'ROLE_ARN', /* required */
solutionVersionArn: 'SOLUTION_VERSION_ARN', /* required */
numResults: 20 /* optional integer*/
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateBatchInferenceJobCommand(createBatchInferenceJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateBatchInferenceJob](#) in *AWS SDK for JavaScript API Reference*.

## Create a batch segment job

The following code example shows how to create a Amazon Personalize batch segment job.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchSegmentJobCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch segment job's parameters.

export const createBatchSegmentJobParam = {
  jobName: 'NAME',
  jobInput: {
    /* required */
    s3DataSource: { /* required */
      path: 'INPUT_PATH', /* required */
      // kmsKeyArn: 'INPUT_KMS_KEY_ARN' /* optional */
    }
  },
  jobOutput: { /* required */
    s3DataDestination: { /* required */
      path: 'OUTPUT_PATH', /* required */
      // kmsKeyArn: 'OUTPUT_KMS_KEY_ARN' /* optional */
    }
  },
  roleArn: 'ROLE_ARN', /* required */
  solutionVersionArn: 'SOLUTION_VERSION_ARN', /* required */
  numResults: 20 /* optional */
};
```

```
export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateBatchSegmentJobCommand(createBatchSegmentJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateBatchSegmentJob](#) in *AWS SDK for JavaScript API Reference*.

## Create a campaign

The following code example shows how to create a Amazon Personalize campaign.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.

import { CreateCampaignCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the campaign's parameters.
export const createCampaignParam = {
  solutionVersionArn: 'SOLUTION_VERSION_ARN', /* required */
  name: 'NAME', /* required */
  minProvisionedTPS: 1 /* optional integer */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateCampaignCommand(createCampaignParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateCampaign](#) in *AWS SDK for JavaScript API Reference*.

## Create a dataset

The following code example shows how to create a Amazon Personalize dataset.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset's parameters.
export const createDatasetParam = {
  datasetGroupArn: 'DATASET_GROUP_ARN', /* required */
  datasetType: 'DATASET_TYPE', /* required */
  name: 'NAME', /* required */
  schemaArn: 'SCHEMA_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateDatasetCommand(createDatasetParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateDataset](#) in *AWS SDK for JavaScript API Reference*.

## Create a dataset export job

The following code example shows how to create a Amazon Personalize dataset export job.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetExportJobCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the export job parameters.
export const datasetExportJobParam = {
  datasetArn: 'DATASET_ARN', /* required */
  jobOutput: {
    s3DataDestination: {
```

```
        path: 'S3_DESTINATION_PATH' /* required */  
        //kmsKeyArn: 'ARN' /* include if your bucket uses AWS KMS for encryption  
    },  
},  
jobName: 'NAME',/* required */  
roleArn: 'ROLE_ARN' /* required */  
}  
  
export const run = async () => {  
    try {  
        const response = await personalizeClient.send(new  
CreateDatasetExportJobCommand(datasetExportJobParam));  
        console.log("Success", response);  
        return response; // For unit tests.  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For API details, see [CreateDatasetExportJob](#) in *AWS SDK for JavaScript API Reference*.

## Create a dataset group

The following code example shows how to create a Amazon Personalize dataset group.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.  
  
import { CreateDatasetGroupCommand } from  
    "@aws-sdk/client-personalize";  
import { personalizeClient } from "./libs/personalizeClients.js";  
  
// Or, create the client here.  
// const personalizeClient = new PersonalizeClient({ region: "REGION" });  
  
// Set the dataset group parameters.  
export const createDatasetGroupParam = {  
    name: 'NAME' /* required */  
}  
  
export const run = async (createDatasetGroupParam) => {  
    try {  
        const response = await personalizeClient.send(new  
CreateDatasetGroupCommand(createDatasetGroupParam));  
        console.log("Success", response);  
        return "Run successfully"; // For unit tests.  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run(createDatasetGroupParam);
```

Create a domain dataset group.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetGroupCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the domain dataset group parameters.
export const domainDatasetGroupParams = {
  name: 'NAME', /* required */
  domain: 'DOMAIN' /* required for a domain dsg, specify ECOMMERCE or VIDEO_ON_DEMAND */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateDatasetGroupCommand(domainDatasetGroupParams));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateDatasetGroup](#) in *AWS SDK for JavaScript API Reference*.

## Create a dataset import job

The following code example shows how to create a Amazon Personalize dataset import job.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import {CreateDatasetImportJobCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset import job parameters.
export const datasetImportJobParam = {
  datasetArn: 'DATASET_ARN', /* required */
  dataSource: { /* required */
    dataLocation: 'S3_PATH'
  },
  jobName: 'NAME',/* required */
  roleArn: 'ROLE_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateDatasetImportJobCommand(datasetImportJobParam));
```

```
    console.log("Success", response);
    return response; // For unit tests.
} catch (err) {
    console.log("Error", err);
}
};

run();
```

- For API details, see [CreateDatasetImportJob](#) in *AWS SDK for JavaScript API Reference*.

## Create a domain schema

The following code example shows how to create a Amazon Personalize domain schema.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from
    "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from 'fs';

let schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
    mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
    mySchema = 'TEST' // for unit tests.
}

// Set the domain schema parameters.
export const createDomainSchemaParam = {
    name: 'NAME', /* required */
    schema: mySchema, /* required */
    domain: 'DOMAIN' /* required for a domain dataset group, specify ECOMMERCE or
VIDEO_ON_DEMAND */
};

export const run = async () => {
    try {
        const response = await personalizeClient.send(new
CreateSchemaCommand(createDomainSchemaParam));
        console.log("Success", response);
        return response; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For API details, see [CreateSchema](#) in *AWS SDK for JavaScript API Reference*.

## Create a filter

The following code example shows how to create a Amazon Personalize filter.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateFilterCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the filter's parameters.
export const createFilterParam = {
  datasetGroupArn: 'DATASET_GROUP_ARN', /* required */
  name: 'NAME', /* required */
  filterExpression: 'FILTER_EXPRESSION' /*required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateFilterCommand(createFilterParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateFilter](#) in *AWS SDK for JavaScript API Reference*.

## Create a recommender

The following code example shows how to create a Amazon Personalize recommender.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateRecommenderCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the recommender's parameters.
export const createRecommenderParam = {
```

```
    name: 'NAME', /* required */
    recipeArn: 'RECIPE_ARN', /* required */
    datasetGroupArn: 'DATASET_GROUP_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateRecommenderCommand(createRecommenderParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};

run();
```

- For API details, see [CreateRecommender](#) in *AWS SDK for JavaScript API Reference*.

## Create a schema

The following code example shows how to create a Amazon Personalize schema.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from 'fs';

let schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
  mySchema = 'TEST' // For unit tests.
}
// Set the schema parameters.
export const createSchemaParam = {
  name: 'NAME', /* required */
  schema: mySchema /* required */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateSchemaCommand(createSchemaParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
```

```
    }
};

run();
```

- For API details, see [CreateSchema](#) in *AWS SDK for JavaScript API Reference*.

## Create a solution

The following code example shows how to create a Amazon Personalize solution.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateSolutionCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the solution parameters.
export const createSolutionParam = {
  datasetGroupArn: 'DATASET_GROUP_ARN', /* required */
  recipeArn: 'RECIPE_ARN', /* required */
  name: 'NAME' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateSolutionCommand(createSolutionParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateSolution](#) in *AWS SDK for JavaScript API Reference*.

## Create a solution version

The following code example shows how to create a Amazon Personalize solution.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateSolutionVersionCommand } from
```

```
    "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the solution version parameters.
export const solutionVersionParam = {
  solutionArn: 'SOLUTION_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateSolutionVersionCommand(solutionVersionParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateSolutionVersion](#) in *AWS SDK for JavaScript API Reference*.

## Create an event tracker

The following code example shows how to create a Amazon Personalize event tracker.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { CreateEventTrackerCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the event tracker's parameters.
export const createEventTrackerParam = {
  datasetGroupArn: 'DATASET_GROUP_ARN', /* required */
  name: 'NAME', /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
CreateEventTrackerCommand(createEventTrackerParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [CreateEventTracker](#) in *AWS SDK for JavaScript API Reference*.

## Import real-time interaction event data

The following code example shows how to import real-time interaction event data into Amazon Personalize Events.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { PutItemsCommand } from
  "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region: "REGION"});

// Set the put items parameters. For string properties and values, use the \ character
// to escape quotes.
var putItemsParam = {
  datasetArn: 'DATASET_ARN', /* required */
  items: [ /* required */
    {
      'itemId': 'ITEM_ID', /* required */
      'properties': "{\"PROPERTY1_NAME\": \"PROPERTY1_VALUE\", \"PROPERTY2_NAME\": \"PROPERTY2_VALUE\", \"PROPERTY3_NAME\": \"PROPERTY3_VALUE\"}"} /* optional */
    }
  ];
};

export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(new
PutItemsCommand(putItemsParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [PutItems](#) in *AWS SDK for JavaScript API Reference*.

## Amazon Personalize Events examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon Personalize Events.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3073\)](#)

## Actions

### Import real-time interaction event data

The following code example shows how to import real-time interaction event data into Amazon Personalize Events.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { PutEventsCommand } from
  "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region: "REGION"});

// Convert your UNIX timestamp to a Date.
const sentAtDate = new Date(1613443801 * 1000) // 1613443801 is a testing value.
Replace it with your sentAt timestamp in UNIX format.

// Set put events parameters.
var putEventsParam = {
  eventList: [           /* required */
    {
      eventType: 'EVENT_TYPE',      /* required */
      sentAt: sentAtDate,          /* required, must be a Date with js */
      eventId: 'EVENT_ID',        /* optional */
      itemId: 'ITEM_ID'          /* optional */
    }
  ],
  sessionId: 'SESSION_ID',      /* required */
  trackingId: 'TRACKING_ID',   /* required */
  userId: 'USER_ID'            /* required */
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(new
PutEventsCommand(putEventsParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [PutEvents](#) in [AWS SDK for JavaScript API Reference](#).

### Incrementally import a user

The following code example shows how to incrementally import a user into Amazon Personalize Events.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { PutUsersCommand } from
  "@aws-sdk/client-personalize-events";
import { personalizeEventsClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeEventsClient = new PersonalizeEventsClient({ region: "REGION"});

// Set the put users parameters. For string properties and values, use the \ character
// to escape quotes.
var putUsersParam = {
  datasetArn: "DATASET_ARN",
  users: [
    {
      'userId': 'USER_ID',
      'properties': "{\"PROPERTY1_NAME\": \"PROPERTY1_VALUE\"}"
    }
  ]
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(new
PutUsersCommand(putUsersParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [PutUsers](#) in *AWS SDK for JavaScript API Reference*.

## Amazon Personalize Runtime examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon Personalize Runtime.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3074\)](#)

## Actions

### Get recommendations (custom dataset group)

The following code example shows how to get Amazon Personalize Runtime ranked recommendations.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { GetPersonalizedRankingCommand } from
  "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region: "REGION"});

// Set the ranking request parameters.
export const getPersonalizedRankingParam = {
  campaignArn: "CAMPAIGN_ARN", /* required */
  userId: 'USER_ID', /* required */
  inputList: ["ITEM_ID_1", "ITEM_ID_2", "ITEM_ID_3", "ITEM_ID_4"]
}

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
      GetPersonalizedRankingCommand(getPersonalizedRankingParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [GetPersonalizedRanking](#) in *AWS SDK for JavaScript API Reference*.

## Get recommendations from a recommender (domain dataset group)

The following code example shows how to get Amazon Personalize Runtime Runtime recommendations.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from
  "@aws-sdk/client-personalize-runtime";

import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region: "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
  campaignArn: 'CAMPAIGN_ARN', /* required */
  userId: 'USER_ID', /* required */
  numResults: 15 /* optional */
}
```

```
export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
GetRecommendationsCommand(getRecommendationsParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Get recommendation with a filter (custom dataset group).

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from
  "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region: "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
  recommenderArn: 'RECOMMENDER_ARN', /* required */
  userId: 'USER_ID', /* required */
  numResults: 15 /* optional */
}

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
GetRecommendationsCommand(getRecommendationsParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Get filtered recommendations from a recommender created in a domain dataset group.

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from
  "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here:
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region: "REGION"});

// Set recommendation request parameters.
export const getRecommendationsParam = {
  campaignArn: 'CAMPAIGN_ARN', /* required */
  userId: 'USER_ID', /* required */
  numResults: 15, /* optional */
  filterArn: 'FILTER_ARN', /* required to filter recommendations */
  filterValues: {
    "PROPERTY": "\"VALUE\""
      /* Only required if your filter has a placeholder
parameter */
  }
}
```

```
export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
GetRecommendationsCommand(getRecommendationsParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [GetRecommendations](#) in *AWS SDK for JavaScript API Reference*.

## Amazon Pinpoint examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon Pinpoint.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3077\)](#)

## Actions

### Send email and text messages

The following code example shows how to send email and text messages with Amazon Pinpoint.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { PinpointClient } from "@aws-sdk/client-pinpoint";
// Set the AWS Region.
const REGION = "us-east-1";
//Set the MediaConvert Service Object
const pinClient = new PinpointClient({region: REGION});
export { pinClient };
```

Send an email message.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessagesCommand } from "@aws-sdk/client-pinpoint";
import { pinClient } from "./libs/pinClient.js";

// The FromAddress must be verified in SES.
const fromAddress = "FROM_ADDRESS";
```

```
const toAddress = "TO_ADDRESS";
const projectId = "PINPOINT_PROJECT_ID";

// The subject line of the email.
var subject = "Amazon Pinpoint Test (AWS SDK for JavaScript in Node.js)";

// The email body for recipients with non-HTML email clients.
var body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)
-----
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in Node.js.
For more information, see https://aws.amazon.com/sdk-for-node-js/`;

// The body of the email for recipients whose email clients support HTML content.
var body_html = `<html>
<head></head>
<body>
  <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>
  <p>This email was sent with
    <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint Email API</a> using
    the
    <a href='https://aws.amazon.com/sdk-for-node-js/'>
      AWS SDK for JavaScript in Node.js</a>.</p>
</body>
</html>`;

// The character encoding for the subject line and message body of the email.
var charset = "UTF-8";

const params = {
  ApplicationId: projectId,
  MessageRequest: {
    Addresses: {
      [toAddress]: {
        ChannelType: "EMAIL",
      },
    },
    MessageConfiguration: {
      EmailMessage: {
        FromAddress: fromAddress,
        SimpleEmail: {
          Subject: {
            Charset: charset,
            Data: subject,
          },
          HtmlPart: {
            Charset: charset,
            Data: body_html,
          },
          TextPart: {
            Charset: charset,
            Data: body_text,
          },
        },
      },
    },
  },
};

const run = async () => {
  try {
    const data = await pinClient.send(new SendMessagesCommand(params));

    const {
      MessageResponse: { Result },
    } = data;
```

```
    const recipientResult = Result[toAddress];

    if (recipientResult.StatusCode !== 200) {
        throw new Error(recipientResult.StatusMessage);
    } else {
        console.log(recipientResult.MessageId);
    }
} catch (err) {
    console.log(err.message);
}
};

run();
```

Send an SMS message.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessagesCommand } from "@aws-sdk/client-pinpoint";
import { pinClient } from "./libs/pinClient.js";

("use strict");

/* The phone number or short code to send the message from. The phone number
or short code that you specify has to be associated with your Amazon Pinpoint
account. For best results, specify long codes in E.164 format. */
const originationNumber = "SENDER_NUMBER"; //e.g., +1XXXXXXXXXX

// The recipient's phone number. For best results, you should specify the phone number
// in E.164 format.
const destinationNumber = "RECEIVER_NUMBER"; //e.g., +1XXXXXXXXXX

// The content of the SMS message.
const message =
    "This message was sent through Amazon Pinpoint " +
    "using the AWS SDK for JavaScript in Node.js. Reply STOP to " +
    "opt out./";

/*The Amazon Pinpoint project/application ID to use when you send this message.
Make sure that the SMS channel is enabled for the project or application
that you choose.*/
const projectId = "PINPOINT_PROJECT_ID"; //e.g., XXXXXXXX66e4e9986478cXXXXXXX

/* The type of SMS message that you want to send. If you plan to send
time-sensitive content, specify TRANSACTIONAL. If you plan to send
marketing-related content, specify PROMOTIONAL.*/
var messageType = "TRANSACTIONAL";

// The registered keyword associated with the originating short code.
var registeredKeyword = "myKeyword";

/* The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html.\*/

var senderId = "MySenderId";

// Specify the parameters to pass to the API.
var params = {
    ApplicationId: projectId,
    MessageRequest: {
        Addresses: [
            [destinationNumber]: {
                ChannelType: "SMS",

```

```
        },
    },
    MessageConfiguration: {
        SMSMessage: {
            Body: message,
            Keyword: registeredKeyword,
            MessageType: messageType,
            OriginationNumber: originationNumber,
            SenderId: senderId,
        },
    },
};

const run = async () => {
    try {
        const data = await pinClient.send(new SendMessagesCommand(params));
        return data; // For unit tests.
        console.log(
            "Message sent! " +
            data["MessageResponse"]["Result"][destinationNumber]["StatusMessage"]
        );
    } catch (err) {
        console.log(err);
    }
};
run();
```

- For API details, see [SendMessages](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Send an email message.

```
'use strict';

const AWS = require('aws-sdk');

// The AWS Region that you want to use to send the email. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/
const aws_region = "us-west-2"

// The "From" address. This address has to be verified in Amazon Pinpoint
// in the region that you use to send email.
const senderAddress = "sender@example.com";

// The address on the "To" line. If your Amazon Pinpoint account is in
// the sandbox, this address also has to be verified.
var toAddress = "recipient@example.com";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
const appId = "ce796be37f32f178af652b26eexample";

// The subject line of the email.
var subject = "Amazon Pinpoint (AWS SDK for JavaScript in Node.js)";
```

```
// The email body for recipients with non-HTML email clients.  
var body_text = `Amazon Pinpoint Test (SDK for JavaScript in Node.js)  
-----  
This email was sent with Amazon Pinpoint using the AWS SDK for JavaScript in Node.js.  
For more information, see https://aws.amazon.com/sdk-for-node-js/`;  
  
// The body of the email for recipients whose email clients support HTML content.  
var body_html = `<html>  
<head></head>  
<body>  
    <h1>Amazon Pinpoint Test (SDK for JavaScript in Node.js)</h1>  
    <p>This email was sent with  
        <a href='https://aws.amazon.com/pinpoint/'>the Amazon Pinpoint API</a> using the  
        <a href='https://aws.amazon.com/sdk-for-node-js/'>  
            AWS SDK for JavaScript in Node.js</a>.</p>  
</body>  
</html>`;  
  
// The character encoding you want to use for the subject line and  
// message body of the email.  
var charset = "UTF-8";  
  
// Specify that you're using a shared credentials file.  
var credentials = new AWS.SharedIniFileCredentials({profile: 'default'});  
AWS.config.credentials = credentials;  
  
// Specify the region.  
AWS.config.update({region:aws_region});  
  
// Create a new Pinpoint object.  
var pinpoint = new AWS.Pinpoint();  
  
// Specify the parameters to pass to the API.  
var params = {  
    ApplicationId: appId,  
    MessageRequest: {  
        Addresses: {  
            [toAddress]:{  
                ChannelType: 'EMAIL'  
            }  
        },  
        MessageConfiguration: {  
            EmailMessage: {  
                FromAddress: senderAddress,  
                SimpleEmail: {  
                    Subject: {  
                        Charset: charset,  
                        Data: subject  
                    },  
                    HtmlPart: {  
                        Charset: charset,  
                        Data: body_html  
                    },  
                    TextPart: {  
                        Charset: charset,  
                        Data: body_text  
                    }  
                }  
            }  
        }  
    }  
};  
  
// Try to send the email.  
pinpoint.sendMessages(params, function(err, data) {  
    // If something goes wrong, print an error message.
```

```
    if(err) {
        console.log(err.message);
    } else {
        console.log("Email sent! Message ID: ", data['MessageResponse']['Result']
[toAddress]['MessageId']);
    }
});
```

Send an SMS message.

```
'use strict';

var AWS = require('aws-sdk');

// The AWS Region that you want to use to send the message. For a list of
// AWS Regions where the Amazon Pinpoint API is available, see
// https://docs.aws.amazon.com/pinpoint/latest/apireference/.
var aws_region = "us-east-1";

// The phone number or short code to send the message from. The phone number
// or short code that you specify has to be associated with your Amazon Pinpoint
// account. For best results, specify long codes in E.164 format.
var originationNumber = "+12065550199";

// The recipient's phone number. For best results, you should specify the
// phone number in E.164 format.
var destinationNumber = "+14255550142";

// The content of the SMS message.
var message = "This message was sent through Amazon Pinpoint "
    + "using the AWS SDK for JavaScript in Node.js. Reply STOP to "
    + "opt out.";

// The Amazon Pinpoint project/application ID to use when you send this message.
// Make sure that the SMS channel is enabled for the project or application
// that you choose.
var applicationId = "ce796be37f32f178af652b26eexample";

// The type of SMS message that you want to send. If you plan to send
// time-sensitive content, specify TRANSACTIONAL. If you plan to send
// marketing-related content, specify PROMOTIONAL.
var messageType = "TRANSACTIONAL";

// The registered keyword associated with the originating short code.
var registeredKeyword = "myKeyword";

// The sender ID to use when sending the message. Support for sender ID
// varies by country or region. For more information, see
// https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html
var senderId = "MySenderId";

// Specify that you're using a shared credentials file, and optionally specify
// the profile that you want to use.
var credentials = new AWS.SharedIniFileCredentials({profile: 'default'});
AWS.config.credentials = credentials;

// Specify the region.
AWS.config.update({region:aws_region});

//Create a new Pinpoint object.
var pinpoint = new AWS.Pinpoint();
```

```
// Specify the parameters to pass to the API.
var params = {
    ApplicationId: applicationId,
    MessageRequest: {
        Addresses: [
            [destinationNumber]: {
                ChannelType: 'SMS'
            }
        ],
        MessageConfiguration: {
            SMSMessage: {
                Body: message,
                Keyword: registeredKeyword,
                MessageType: messageType,
                OriginationNumber: originationNumber,
                SenderId: senderId,
            }
        }
    }
};

//Try to send the message.
pinpoint.sendMessages(params, function(err, data) {
    // If something goes wrong, print an error message.
    if(err) {
        console.log(err.message);
    // Otherwise, show the unique ID for the message.
    } else {
        console.log("Message sent! "
            + data['MessageResponse']['Result'][destinationNumber]['StatusMessage']);
    }
});
```

- For API details, see [SendMessages](#) in *AWS SDK for JavaScript API Reference*.

## Amazon Redshift examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon Redshift.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3083\)](#)

## Actions

### Create a cluster

The following code example shows how to create an Amazon Redshift cluster.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { RedshiftClient } = require("@aws-sdk/client-redshift");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Create the cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

const params = {
    ClusterIdentifier: "CLUSTER_NAME", // Required
    NodeType: "NODE_TYPE", //Required
    MasterUsername: "MASTER_USER_NAME", // Required - must be lowercase
    MasterUserPassword: "MASTER_USER_PASSWORD", // Required - must contain at least one
    uppercase letter, and one number
    ClusterType: "CLUSTER_TYPE", // Required
    IAMRoleARN: "IAM_ROLE_ARN", // Optional - the ARN of an IAM role with permissions
    your cluster needs to access other AWS services on your behalf, such as Amazon S3.
    ClusterSubnetGroupName: "CLUSTER_SUBNET_GROUPNAME", //Optional - the name of a
    cluster subnet group to be associated with this cluster. Defaults to 'default' if not
    specified.
    DBName: "DATABASE_NAME", // Optional - defaults to 'dev' if not specified
    Port: "PORT_NUMBER", // Optional - defaults to '5439' if not specified
};

const run = async () => {
    try {
        const data = await redshiftClient.send(new CreateClusterCommand(params));
        console.log(
            "Cluster " + data.Cluster.ClusterIdentifier + " successfully created"
        );
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For API details, see [CreateCluster](#) in *AWS SDK for JavaScript API Reference*.

## Delete a cluster

The following code example shows how to delete an Amazon Redshift cluster.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { RedshiftClient } = require("@aws-sdk/client-redshift");
```

```
// Set the AWS Region.  
const REGION = "REGION";  
//Set the Redshift Service Object  
const redshiftClient = new RedshiftClient({ region: REGION });  
export { redshiftClient };
```

Create the cluster.

```
// Import required AWS SDK clients and commands for Node.js  
import { DeleteClusterCommand } from "@aws-sdk/client-redshift";  
import { redshiftClient } from "./libs/redshiftClient.js";  
  
const params = {  
    ClusterIdentifier: "CLUSTER_NAME",  
    SkipFinalClusterSnapshot: false,  
    FinalClusterSnapshotIdentifier: "CLUSTER_SNAPSHOT_ID",  
};  
  
const run = async () => {  
    try {  
        const data = await redshiftClient.send(new DeleteClusterCommand(params));  
        console.log("Success, cluster deleted. ", data);  
        return data; // For unit tests.  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For API details, see [DeleteCluster](#) in *AWS SDK for JavaScript API Reference*.

## Describe your clusters

The following code example shows how to describe your Amazon Redshift clusters.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { RedshiftClient } = require("@aws-sdk/client-redshift");  
// Set the AWS Region.  
const REGION = "REGION";  
//Set the Redshift Service Object  
const redshiftClient = new RedshiftClient({ region: REGION });  
export { redshiftClient };
```

Describe your clusters.

```
// Import required AWS SDK clients and commands for Node.js  
import { DescribeClustersCommand } from "@aws-sdk/client-redshift";  
import { redshiftClient } from "./libs/redshiftClient.js";  
  
const params = {  
    ClusterIdentifier: "CLUSTER_NAME",
```

```
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new DescribeClustersCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [DescribeClusters in AWS SDK for JavaScript API Reference](#).

## Modify a cluster

The following code example shows how to modify an Amazon Redshift cluster.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { RedshiftClient } = require("@aws-sdk/client-redshift");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const redshiftClient = new RedshiftClient({ region: REGION });
export { redshiftClient };
```

Modify a cluster.

```
// Import required AWS SDK clients and commands for Node.js
import { ModifyClusterCommand } from "@aws-sdk/client-redshift";
import { redshiftClient } from "./libs/redshiftClient.js";

// Set the parameters
const params = {
  ClusterIdentifier: "CLUSTER_NAME",
  MasterUserPassword: "NEW_MASTER_USER_PASSWORD",
};

const run = async () => {
  try {
    const data = await redshiftClient.send(new ModifyClusterCommand(params));
    console.log("Success was modified.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [ModifyCluster in AWS SDK for JavaScript API Reference](#).

## Amazon S3 examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon Simple Storage Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3087\)](#)
- [Scenarios \(p. 3105\)](#)

## Actions

### Add CORS rules to a bucket

The following code example shows how to add cross-origin resource sharing (CORS) rules to an S3 bucket.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Add a CORS rule.

```
// Import required AWS-SDK clients and commands for Node.js.  
import { PutBucketCorsCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an  
// Amazon S3 service client module.  
  
// Set parameters.  
// Create initial parameters JSON for putBucketCors.  
const thisConfig = {  
    AllowedHeaders: ["Authorization"],  
    AllowedMethods: [],  
    AllowedOrigins: ["*"],  
    ExposeHeaders: [],  
    MaxAgeSeconds: 3000,  
};  
  
// Assemble the list of allowed methods based on command line parameters  
const allowedMethods = [];  
process.argv.forEach(function (val) {
```

```
if (val.toUpperCase() === "POST") {
    allowedMethods.push("POST");
}
if (val.toUpperCase() === "GET") {
    allowedMethods.push("GET");
}
if (val.toUpperCase() === "PUT") {
    allowedMethods.push("PUT");
}
if (val.toUpperCase() === "PATCH") {
    allowedMethods.push("PATCH");
}
if (val.toUpperCase() === "DELETE") {
    allowedMethods.push("DELETE");
}
if (val.toUpperCase() === "HEAD") {
    allowedMethods.push("HEAD");
}
});

// Copy the array of allowed methods into the config object
thisConfig.AllowedMethods = allowedMethods;

// Create an array of configs then add the config object to it.
const corsRules = new Array(thisConfig);

// Create CORS parameters.
export const corsParams = {
    Bucket: "BUCKET_NAME",
    CORSConfiguration: { CORSRules: corsRules },
};

export async function run() {
    try {
        const data = await s3Client.send(new PutBucketCorsCommand(corsParams));
        console.log("Success", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
}
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutBucketCors](#) in [AWS SDK for JavaScript API Reference](#).

## Add a policy to a bucket

The following code example shows how to add a policy to an S3 bucket.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
```

```
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Add the policy.

```
// Import required AWS SDK clients and commands for Node.js.  
import { CreateBucketCommand, PutBucketPolicyCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an  
Amazon S3 service client module.  
  
const BUCKET_NAME = "BUCKET_NAME";  
export const bucketParams = {  
    Bucket: BUCKET_NAME,  
};  
// Create the policy in JSON for the S3 bucket.  
const readOnlyAnonUserPolicy = {  
    Version: "2012-10-17",  
    Statement: [  
        {  
            Sid: "AddPerm",  
            Effect: "Allow",  
            Principal: "*",  
            Action: ["s3:GetObject"],  
            Resource: ["*"],  
        },  
    ],  
};  
  
// Create selected bucket resource string for bucket policy.  
const bucketResource = "arn:aws:s3:::" + BUCKET_NAME + "/*"; //BUCKET_NAME  
readOnlyAnonUserPolicy.Statement[0].Resource[0] = bucketResource;  
  
// Convert policy JSON into string and assign into parameters.  
const bucketPolicyParams = {  
    Bucket: BUCKET_NAME,  
    Policy: JSON.stringify(readOnlyAnonUserPolicy),  
};  
  
export const run = async () => {  
    try {  
        const data = await s3Client.send(  
            new CreateBucketCommand(bucketParams)  
        );  
        console.log('Success, bucket created.', data)  
        try {  
            const response = await s3Client.send(  
                new PutBucketPolicyCommand(bucketPolicyParams)  
            );  
            console.log("Success, permissions added to bucket", response);  
            return response;  
        }  
        catch (err) {  
            console.log("Error adding policy to S3 bucket.", err);  
        }  
    } catch (err) {  
        console.log("Error creating S3 bucket.", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [PutBucketPolicy](#) in *AWS SDK for JavaScript API Reference*.

## Copy an object from one bucket to another

The following code example shows how to copy an S3 object from one bucket to another.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Copy the object.

```
// Get service clients module and commands using ES6 syntax.  
import { CopyObjectCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js";  
  
// Set the bucket parameters.  
  
export const params = {  
    Bucket: "DESTINATION_BUCKET_NAME",  
    CopySource: "/SOURCE_BUCKET_NAME/OBJECT_NAME",  
    Key: "OBJECT_NAME"  
};  
  
// Create the Amazon S3 bucket.  
export const run = async () => {  
    try {  
        const data = await s3Client.send(new CopyObjectCommand(params));  
        console.log("Success", data);  
        return data; // For unit tests.  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For API details, see [CopyObject](#) in *AWS SDK for JavaScript API Reference*.

## Create a bucket

The following code example shows how to create an S3 bucket.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Create the bucket.

```
// Get service clients module and commands using ES6 syntax.  
import { CreateBucketCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js";  
  
// Set the bucket parameters.  
  
export const bucketParams = { Bucket: "BUCKET_NAME" };  
  
// Create the Amazon S3 bucket.  
export const run = async () => {  
    try {  
        const data = await s3Client.send(new CreateBucketCommand(bucketParams));  
        console.log("Success", data);  
        return data; // For unit tests.  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateBucket](#) in [AWS SDK for JavaScript API Reference](#).

## Delete a policy from a bucket

The following code example shows how to delete a policy from an S3 bucket.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Delete the bucket policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { DeleteBucketPolicyCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Set the bucket parameters
export const bucketParams = { Bucket: "BUCKET_NAME" };

export const run = async () => {
  try {
    const data = await s3Client.send(new DeleteBucketPolicyCommand(bucketParams));
    console.log("Success", data + ", bucket policy deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
// Invoke run() so these examples run out of the box.
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteBucketPolicy](#) in [AWS SDK for JavaScript API Reference](#).

## Delete an empty bucket

The following code example shows how to delete an empty S3 bucket.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Delete the bucket.

```
// Import required AWS SDK clients and commands for Node.js.
import { DeleteBucketCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Set the bucket parameters
export const bucketParams = { Bucket: "BUCKET_NAME" };

export const run = async () => {
  try {
    const data = await s3Client.send(new DeleteBucketCommand(bucketParams));
    return data; // For unit tests.
  }
```

```
        console.log("Success - bucket deleted");
    } catch (err) {
        console.log("Error", err);
    }
};

// Invoke run() so these examples run out of the box.
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteBucket](#) in [AWS SDK for JavaScript API Reference](#).

## Delete an object

The following code example shows how to delete an S3 object.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Delete an object.

```
import { DeleteObjectCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js" // Helper function that creates an Amazon
S3 service client module.

export const bucketParams = { Bucket: "BUCKET_NAME", Key: "KEY" };

export const run = async () => {
    try {
        const data = await s3Client.send(new DeleteObjectCommand(bucketParams));
        console.log("Success. Object deleted.", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For API details, see [DeleteObject](#) in [AWS SDK for JavaScript API Reference](#).

## Delete multiple objects

The following code example shows how to delete multiple objects from an S3 bucket.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Delete multiple objects.

```
import { DeleteObjectsCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js" // Helper function that creates an Amazon  
S3 service client module.  
  
export const bucketParams = {  
    Bucket: "BUCKET_NAME",  
    Delete: {  
        Objects: [  
            {  
                Key: "KEY_1",  
            },  
            {  
                Key: "KEY_2",  
            },  
        ],  
    },  
};  
  
export const run = async () => {  
    try {  
        const data = await s3Client.send(new DeleteObjectsCommand(bucketParams));  
        return data; // For unit tests.  
        console.log("Success. Object deleted.");  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

Delete all objects in a bucket.

```
import { ListObjectsCommand, DeleteObjectCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an Amazon  
S3 service client module.  
  
export const bucketParams = { Bucket: "BUCKET_NAME" };  
  
export const run = async () => {  
    try {  
        const data = await s3Client.send(new ListObjectsCommand(bucketParams));  
        let noOfObjects = data.Contents;  
        for (let i = 0; i < noOfObjects.length; i++) {  
            await s3Client.send(  
                new DeleteObjectCommand({  
                    Bucket: "BUCKET_NAME",  
                    Key: data.Contents[i].Key,  
                })  
            );  
        }  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

```
        new DeleteObjectCommand({
          Bucket: bucketParams.Bucket,
          Key: noOfObjects[i].Key,
        })
      );
    }
    console.log("Success. Objects deleted.");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};

run();
```

- For API details, see [DeleteObjects](#) in *AWS SDK for JavaScript API Reference*.

## Delete the website configuration from a bucket

The following code example shows how to delete the website configuration from an S3 bucket.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Delete the website configuration from the bucket.

```
// Import required AWS SDK clients and commands for Node.js.

import { DeleteBucketWebsiteCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Create the parameters for calling
export const bucketParams = { Bucket: "BUCKET_NAME" };

export const run = async () => {
  try {
    const data = await s3Client.send(new DeleteBucketWebsiteCommand(bucketParams));
    return data; // For unit tests.
    console.log("Success", data);
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteBucketWebsite](#) in [AWS SDK for JavaScript API Reference](#).

## Get CORS rules for a bucket

The following code example shows how to get cross-origin resource sharing (CORS) rules for an S3 bucket.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Get the CORS policy for the bucket.

```
// Import required AWS SDK clients and commands for Node.js.  
import { GetBucketCorsCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an  
Amazon S3 service client module.  
  
// Create the parameters for calling  
export const bucketParams = { Bucket: "BUCKET_NAME" };  
  
export const run = async () => {  
    try {  
        const data = await s3Client.send(new GetBucketCorsCommand(bucketParams));  
        console.log("Success", JSON.stringify(data.CORSRules));  
        return data; // For unit tests.  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetBucketCors](#) in [AWS SDK for JavaScript API Reference](#).

## Get an object from a bucket

The following code example shows how to read data from an object in an S3 bucket.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Download the object.

```
// Import required AWS SDK clients and commands for Node.js.  
import { GetObjectCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an  
Amazon S3 service client module.  
  
export const bucketParams = {  
    Bucket: "BUCKET_NAME",  
    Key: "KEY",  
};  
  
export const run = async () => {  
    try {  
        // Get the object} from the Amazon S3 bucket. It is returned as a ReadableStream.  
        const data = await s3Client.send(new GetObjectCommand(bucketParams));  
        // Convert the ReadableStream to a string.  
        return await data.Body.transformToString();  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetObject](#) in [AWS SDK for JavaScript API Reference](#).

## Get the ACL of a bucket

The following code example shows how to get the access control list (ACL) of an S3 bucket.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

### Get the ACL permissions.

```
// Import required AWS SDK clients and commands for Node.js.
import { GetBucketAclCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Create the parameters.
export const bucketParams = { Bucket: "BUCKET_NAME" };

export const run = async () => {
  try {
    const data = await s3Client.send(new GetBucketAclCommand(bucketParams));
    console.log("Success", data.Grants);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetBucketAcl](#) in [AWS SDK for JavaScript API Reference](#).

### Get the policy for a bucket

The following code example shows how to get the policy for an S3 bucket.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Get the bucket policy.

```
// Import required AWS SDK clients and commands for Node.js.
import { GetBucketPolicyCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Create the parameters for calling
export const bucketParams = { Bucket: "BUCKET_NAME" };

export const run = async () => {
  try {
    const data = await s3Client.send(new GetBucketPolicyCommand(bucketParams));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
```

```
    } catch (err) {
      console.log("Error", err);
    }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetBucketPolicy](#) in [AWS SDK for JavaScript API Reference](#).

## Get the website configuration for a bucket

The following code example shows how to get the website configuration for an S3 bucket.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Get the website configuration.

```
// Import required AWS SDK clients and commands for Node.js.
import { GetBucketWebsiteCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Create the parameters for calling
export const bucketParams = { Bucket: "BUCKET_NAME" };

export const run = async () => {
  try {
    const data = await s3Client.send(new GetBucketWebsiteCommand(bucketParams));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For API details, see [GetBucketWebsite](#) in [AWS SDK for JavaScript API Reference](#).

## List buckets

The following code example shows how to list S3 buckets.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

List the buckets.

```
// Import required AWS SDK clients and commands for Node.js.  
import { ListBucketsCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an  
Amazon S3 service client module.  
  
export const run = async () => {  
    try {  
        const data = await s3Client.send(new ListBucketsCommand({}));  
        console.log("Success", data.Buckets);  
        return data; // For unit tests.  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListBuckets](#) in [AWS SDK for JavaScript API Reference](#).

## List objects in a bucket

The following code example shows how to list objects in an S3 bucket.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

List the objects.

```
// Import required AWS SDK clients and commands for Node.js.
import { ListObjectsCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Create the parameters for the bucket
export const bucketParams = { Bucket: "BUCKET_NAME" };

export const run = async () => {
  try {
    const data = await s3Client.send(new ListObjectsCommand(bucketParams));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

List 1000 or more objects.

```
// Import required AWS SDK clients and commands for Node.js.
import { ListObjectsCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

// Create the parameters for the bucket
export const bucketParams = { Bucket: "BUCKET_NAME" };

export async function run() {
  // Declare truncated as a flag that the while loop is based on.
  let truncated = true;
  // Declare a variable to which the key of the last element is assigned to in the
  response.
  let pageMarker;
  // while loop that runs until 'response.truncated' is false.
  while (truncated) {
    try {
      const response = await s3Client.send(new ListObjectsCommand(bucketParams));
      // return response; //For unit tests
      response.Contents.forEach((item) => {
        console.log(item.Key);
      });
      // Log the key of every item in the response to standard output.
      truncated = response.IsTruncated;
      // If truncated is true, assign the key of the last element in the response to
      the pageMarker variable.
      if (truncated) {
        pageMarker = response.Contents.slice(-1)[0].Key;
        // Assign the pageMarker value to bucketParams so that the next iteration
        starts from the new pageMarker.
        bucketParams.Marker = pageMarker;
      }
      // At end of the list, response.truncated is false, and the function exits the
      while loop.
    } catch (err) {
      console.log("Error", err);
      truncated = false;
    }
  }
}
```

```
run();
```

- For API details, see [ListObjects](#) in *AWS SDK for JavaScript API Reference*.

## Set a new ACL for a bucket

The following code example shows how to set a new access control list (ACL) for an S3 bucket.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Put the bucket ACL.

```
// Import required AWS SDK clients and commands for Node.js.  
import { PutBucketAclCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an  
Amazon S3 service client module.  
  
// Set the parameters. For more information,  
// see https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/S3.html#putBucketAcl-property.  
export const bucketParams = {  
  Bucket: "BUCKET_NAME",  
  // 'GrantFullControl' allows grantee the read, write, read ACP, and write ACL  
  // permissions on the bucket.  
  // Use a canonical user ID for an AWS account, formatted as follows:  
  // id=002160194XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXa7a49125274  
  GrantFullControl: "GRANTEE_1",  
  // 'GrantWrite' allows grantee to create, overwrite, and delete any object in the  
  // bucket.  
  // For example, 'uri=http://acs.amazonaws.com/groups/s3/LogDelivery'  
  GrantWrite: "GRANTEE_2",  
};  
  
export const run = async () => {  
  try {  
    const data = await s3Client.send(new PutBucketAclCommand(bucketParams));  
    console.log("Success, permissions added to bucket", data);  
    return data; // For unit tests.  
  } catch (err) {  
    console.log("Error", err);  
  }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [PutBucketAcl](#) in *AWS SDK for JavaScript API Reference*.

## Set the website configuration for a bucket

The following code example shows how to set the website configuration for an S3 bucket.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Set the website configuration.

```
// Import required AWS SDK clients and commands for Node.js.  
import { PutBucketWebsiteCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an  
Amazon S3 service client module.  
  
// Create the parameters for the bucket  
export const bucketParams = { Bucket: "BUCKET_NAME" };  
export const staticHostParams = {  
    Bucket: bucketParams,  
    WebsiteConfiguration: {  
        ErrorDocument: {  
            Key: "",  
        },  
        IndexDocument: {  
            Suffix: "",  
        },  
    },  
};  
  
export const run = async () => {  
    // Insert specified bucket name and index and error documents into parameters JSON  
    // from command line arguments  
    staticHostParams.Bucket = bucketParams;  
    staticHostParams.WebsiteConfiguration.IndexDocument.Suffix = "INDEX_PAGE"; // The  
index document inserted into parameters JSON.  
    staticHostParams.WebsiteConfiguration.ErrorDocument.Key = "ERROR_PAGE"; // The error  
document inserted into parameters JSON.  
    // Set the new website configuration on the selected bucket.  
    try {  
        const data = await s3Client.send(new PutBucketWebsiteCommand(staticHostParams));  
        console.log("Success", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutBucketWebsite](#) in [AWS SDK for JavaScript API Reference](#).

## Upload an object to a bucket

The following code example shows how to upload an object to an S3 bucket.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.  
import { S3Client } from "@aws-sdk/client-s3";  
// Set the AWS Region.  
const REGION = "us-east-1";  
// Create an Amazon S3 service client object.  
const s3Client = new S3Client({ region: REGION });  
export { s3Client };
```

Create and upload the object.

```
// Import required AWS SDK clients and commands for Node.js.  
import { PutObjectCommand } from "@aws-sdk/client-s3";  
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an  
Amazon S3 service client module.  
  
// Set the parameters.  
export const bucketParams = {  
    Bucket: "BUCKET_NAME",  
    // Specify the name of the new object. For example, 'index.html'.  
    // To create a directory for the object, use '/'. For example, 'myApp/package.json'.  
    Key: "OBJECT_NAME",  
    // Content of the new object.  
    Body: "BODY",  
};  
  
// Create and upload the object to the S3 bucket.  
export const run = async () => {  
    try {  
        const data = await s3Client.send(new PutObjectCommand(bucketParams));  
        return data; // For unit tests.  
        console.log(  
            "Successfully uploaded object: " +  
            bucketParams.Bucket +  
            "/" +  
            bucketParams.Key  
        );  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

Upload the object.

```
// Import required AWS SDK clients and commands for Node.js.
import { PutObjectCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
// Amazon S3 service client module.
import {path} from "path";
import {fs} from "fs";

const file = "OBJECT_PATH_AND_NAME"; // Path to and name of object. For example '../
myFiles/index.js'.
const fileStream = fs.createReadStream(file);

// Set the parameters
export const uploadParams = {
  Bucket: "BUCKET_NAME",
  // Add the required 'Key' parameter using the 'path' module.
  Key: path.basename(file),
  // Add the required 'Body' parameter
  Body: fileStream,
};

// Upload file to specified bucket.
export const run = async () => {
  try {
    const data = await s3Client.send(new PutObjectCommand(uploadParams));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [PutObject](#) in [AWS SDK for JavaScript API Reference](#).

## Scenarios

### Create a presigned URL

The following code example shows how to create a presigned URL for S3 and upload an object.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
// Create service client module using ES6 syntax.
import { S3Client } from "@aws-sdk/client-s3";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an Amazon S3 service client object.
const s3Client = new S3Client({ region: REGION });
export { s3Client };
```

Create a presigned URL to upload an object to a bucket.

```
// Import the required AWS SDK clients and commands for Node.js
import {
  CreateBucketCommand,
  DeleteObjectCommand,
  PutObjectCommand,
  DeleteBucketCommand
} from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
// Amazon S3 service client module.
import { getSignedUrl } from "@aws-sdk/s3-request-presigner";
import fetch from "node-fetch";

// Set parameters
// Create a random name for the Amazon Simple Storage Service (Amazon S3) bucket and
key
export const bucketParams = {
  Bucket: `test-bucket-${Math.ceil(Math.random() * 10 ** 10)}`,
  Key: `test-object-${Math.ceil(Math.random() * 10 ** 10)}`,
  Body: "BODY"
};
export const run = async () => {
  try {
    // Create an S3 bucket.
    console.log(`Creating bucket ${bucketParams.Bucket}`);
    await s3Client.send(new CreateBucketCommand({ Bucket: bucketParams.Bucket }));
    console.log(`Waiting for "${bucketParams.Bucket}" bucket creation...`);
  } catch (err) {
    console.log("Error creating bucket", err);
  }
  try {
    // Create a command to put the object in the S3 bucket.
    const command = new PutObjectCommand(bucketParams);
    // Create the presigned URL.
    const signedUrl = await getSignedUrl(s3Client, command, {
      expiresIn: 3600,
    });
    console.log(
      `\nPutting "${bucketParams.Key}" using signedUrl with body "${bucketParams.Body}"` +
      " in v3"
    );
    console.log(signedUrl);
    const response = await fetch(signedUrl, {method: 'PUT', body: bucketParams.Body});
    console.log(
      `\nResponse returned by signed URL: ${await response.text()}\n`
    );
  } catch (err) {
    console.log("Error creating presigned URL", err);
  }
  try {
    // Delete the object.
    console.log(`\nDeleting object "${bucketParams.Key}" from bucket`);
    await s3Client.send(
      new DeleteObjectCommand({ Bucket: bucketParams.Bucket, Key: bucketParams.Key })
    );
  } catch (err) {
    console.log("Error deleting object", err);
  }
  try {
    // Delete the S3 bucket.
    console.log(`\nDeleting bucket ${bucketParams.Bucket}`);
    await s3Client.send(
```

```
        new DeleteBucketCommand({ Bucket: bucketParams.Bucket })
    );
} catch (err) {
    console.log("Error deleting bucket", err);
}
};

run();
```

Create a presigned URL to download an object from a bucket.

```
// Import the required AWS SDK clients and commands for Node.js
import {
    CreateBucketCommand,
    PutObjectCommand,
    GetObjectCommand,
    DeleteObjectCommand,
    DeleteBucketCommand
} from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js"; // Helper function that creates an
// Amazon S3 service client module.
import { getSignedUrl } from "@aws-sdk/s3-request-presigner";
const fetch = require("node-fetch");

// Set parameters
// Create a random names for the S3 bucket and key.
export const bucketParams = {
    Bucket: `test-bucket-${Math.ceil(Math.random() * 10 ** 10)}`,
    Key: `test-object-${Math.ceil(Math.random() * 10 ** 10)}`,
    Body: "BODY"
};

export const run = async () => {
    // Create an S3 bucket.
    try {
        console.log(`Creating bucket ${bucketParams.Bucket}`);
        const data = await s3Client.send(
            new CreateBucketCommand({ Bucket: bucketParams.Bucket })
        );
        return data; // For unit tests.
        console.log(`Waiting for "${bucketParams.Bucket}" bucket creation...\n`);
    } catch (err) {
        console.log("Error creating bucket", err);
    }
    // Put the object in the S3 bucket.
    try {
        console.log(`Putting object "${bucketParams.Key}" in bucket`);
        const data = await s3Client.send(
            new PutObjectCommand({
                Bucket: bucketParams.Bucket,
                Key: bucketParams.Key,
                Body: bucketParams.Body,
            })
        );
        return data; // For unit tests.
    } catch (err) {
        console.log("Error putting object", err);
    }
    // Create a presigned URL.
    try {
        // Create the command.
        const command = new GetObjectCommand(bucketParams);

        // Create the presigned URL.
        const signedUrl = await getSignedUrl(s3Client, command, {
```

```
        expiresIn: 3600,
    });
    console.log(
        `\nGetting "${bucketParams.Key}" using signedUrl with body "${bucketParams.Body}"`  

    in v3
);
    console.log(signedUrl);
    const response = await fetch(signedUrl);
    console.log(
        `\nResponse returned by signed URL: ${await response.text()}\n`  

    );
} catch (err) {
    console.log("Error creating presigned URL", err);
}
// Delete the object.
try {
    console.log(`\nDeleting object "${bucketParams.Key}" from bucket`);
    const data = await s3Client.send(
        new DeleteObjectCommand({ Bucket: bucketParams.Bucket, Key: bucketParams.Key })
    );
    return data; // For unit tests.
} catch (err) {
    console.log("Error deleting object", err);
}
// Delete the S3 bucket.
try {
    console.log(`\nDeleting bucket ${bucketParams.Bucket}`);
    const data = await s3Client.send(
        new DeleteBucketCommand({ Bucket: bucketParams.Bucket, Key: bucketParams.Key })
    );
    return data; // For unit tests.
} catch (err) {
    console.log("Error deleting object", err);
}
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

## Get started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.
- Delete a bucket.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import {
```

```
CreateBucketCommand,
PutObjectCommand,
CopyObjectCommand,
DeleteObjectCommand,
DeleteBucketCommand,
GetObjectCommand
} from "@aws-sdk/client-s3";
import { s3Client } from "../libs/s3Client.js"; // Helper function that creates an
Amazon S3 service client module.

if (process.argv.length < 5) {
    console.log(
        "Usage: node s3_basics.js <the bucket name> <the AWS Region to use> <object name>
<object content>\n" +
        "Example: node s3_basics_full.js test-bucket 'test.txt' 'Test Content'"
    );
}
const bucket_name = process.argv[2];
const object_key = process.argv[3];
const object_content = process.argv[4];

export const run = async (bucket_name, object_key, object_content) => {
    try {
        const create_bucket_params = {
            Bucket: bucket_name
        };
        console.log("\nCreating the bucket, named " + bucket_name + "...\\n");
        console.log("about to create");
        const data = await s3Client.send(
            new CreateBucketCommand(create_bucket_params)
        );
        console.log("Bucket created at ", data.Location);
        try {
            console.log(
                "\nCreated and uploaded an object named " +
                object_key +
                " to first bucket " +
                bucket_name +
                "...\\n"
            );
            // Set the parameters for the object to upload.
            const object_upload_params = {
                Bucket: bucket_name,
                // Specify the name of the new object. For example, 'test.html'.
                // To create a directory for the object, use '/'. For example, 'myApp/
                package.json'.
                Key: object_key,
                // Content of the new object.
                Body: object_content,
            };
            // Create and upload the object to the first S3 bucket.
            await s3Client.send(new PutObjectCommand(object_upload_params));
            console.log(
                "Successfully uploaded object: " +
                object_upload_params.Bucket +
                "/" +
                object_upload_params.Key
            );
            try {
                const download_bucket_params = {
                    Bucket: bucket_name,
                    Key: object_key
                };
                console.log(
                    "\nDownloading " +
                    object_key +
                    " from bucket " +
                    bucket_name
                );
                const downloadObjectParams = {
                    Bucket: bucket_name,
                    Key: object_key
                };
                const downloadObjectCommand = new GetObjectCommand(
                    downloadObjectParams
                );
                const downloadObjectResult = await s3Client.send(
                    downloadObjectCommand
                );
                const fileContent = downloadObjectResult.Body?.getRawBody();
                if (fileContent) {
                    const fileContentString = fileContent.toString();
                    console.log(`Downloaded object content: ${fileContentString}`);
                }
            }
        }
    }
}
```

```
        " from" +
        bucket_name +
        "...\\n"
    );
// Create a helper function to convert a ReadableStream into a string.
const streamToString = (stream) =>
    new Promise((resolve, reject) => {
        const chunks = [];
        stream.on("data", (chunk) => chunks.push(chunk));
        stream.on("error", reject);
        stream.on("end", () => resolve(Buffer.concat(chunks).toString("utf8")));
    });

// Get the object from the Amazon S3 bucket. It is returned as a
ReadableStream.
const data = await s3Client.send(newGetObjectCommand(download_bucket_params));
// Convert the ReadableStream to a string.
const bodyContents = await streamToString(data.Body);
console.log(bodyContents);
try {
    // Copy the object from the first bucket to the second bucket.
    const copy_object_params = {
        Bucket: bucket_name,
        CopySource: "/" + bucket_name + "/" + object_key,
        Key: "copy-destination/" + object_key,
    };
    console.log(
        "\nCopying " +
        object_key +
        " from" +
        bucket_name +
        " to " +
        bucket_name +
        "/" +
        copy_object_params.Key +
        "...\\n"
    );
    await s3Client.send(newCopyObjectCommand(copy_object_params));
    console.log("Success, object copied to folder.");
    try {
        console.log("\nDeleting " + object_key + " from" + bucket_name);
        const delete_object_from_bucket_params = {
            Bucket: bucket_name,
            Key: object_key,
        };
        await s3Client.send(
            newDeleteObjectCommand(delete_object_from_bucket_params)
        );
        console.log("Success. Object deleted from bucket.");
        try {
            console.log(
                "\nDeleting " +
                object_key +
                " from " +
                bucket_name +
                "/copy-destination folder"
            );
            const delete_object_from_folder_params = {
                Bucket: bucket_name,
                Key: "copy-destination/" + object_key,
            };
            await s3Client.send(
                newDeleteObjectCommand(delete_object_from_folder_params)
            );
        }
    }
}
```

```
        console.log("Success. Object deleted from folder.");
        try {
            console.log(
                "\nDeleting the bucket named " + bucket_name + "...\\n"
            );
            const delete_bucket_params = {Bucket: bucket_name};
            await s3Client.send(
                new DeleteBucketCommand(delete_bucket_params)
            );
            console.log("Success. First bucket deleted.");
            return "Run successfully"; // For unit tests.
        } catch (err) {
            console.log("Error deleting object from folder.", err);
            process.exit(1);
        }
    } catch (err) {
        console.log("Error deleting bucket.", err);
        process.exit(1);
    }
} catch (err) {
    console.log("Error deleting object from bucket.", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error copying object from to folder", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error downloading object", err);
    process.exit(1);
}

}
}catch (err) {
    console.log("Error creating and upload object to bucket", err);
    process.exit(1);
}
console.log("works");
} catch (err) {
    console.log("Error creating bucket", err);
}
};

run(bucket_name, object_key, object_content);
```

- For API details, see the following topics in *AWS SDK for JavaScript API Reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

## S3 Glacier examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon S3 Glacier.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3112\)](#)

## Actions

### Create a vault

The following code example shows how to create an Amazon S3 Glacier vault.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

Create the vault.

```
// Load the SDK for JavaScript
import { CreateVaultCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME
const params = { vaultName: vaultname };

const run = async () => {
  try {
    const data = await glacierClient.send(new CreateVaultCommand(params));
    console.log("Success, vault created!");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error");
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateVault](#) in [AWS SDK for JavaScript API Reference](#).

#### SDK for JavaScript V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the SDK for JavaScript
```

```
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create a new service object
var glacier = new AWS.Glacier({apiVersion: '2012-06-01'});
// Call Glacier to create the vault
glacier.createVault({vaultName: 'YOUR_VAULT_NAME'}, function(err) {
    if (!err) {
        console.log("Created vault!")
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateVault](#) in [AWS SDK for JavaScript API Reference](#).

## Upload an archive to a vault

The following code example shows how to upload an archive to an Amazon S3 Glacier vault.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { GlacierClient } = require("@aws-sdk/client-glacier");
// Set the AWS Region.
const REGION = "REGION";
//Set the Redshift Service Object
const glacierClient = new GlacierClient({ region: REGION });
export { glacierClient };
```

Upload the archive.

```
// Load the SDK for JavaScript
import { UploadArchiveCommand } from "@aws-sdk/client-glacier";
import { glacierClient } from "./libs/glacierClient.js";

// Set the parameters
const vaultname = "VAULT_NAME"; // VAULT_NAME

// Create a new service object and buffer
const buffer = new Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer
const params = { vaultName: vaultname, body: buffer };

const run = async () => {
    try {
        const data = await glacierClient.send(new UploadArchiveCommand(params));
        console.log("Archive ID", data.archiveId);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error uploading archive!", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UploadArchive](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the SDK for JavaScript
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create a new service object and buffer
var glacier = new AWS.Glacier({apiVersion: '2012-06-01'});
buffer = Buffer.alloc(2.5 * 1024 * 1024); // 2.5MB buffer

var params = {vaultName: 'YOUR_VAULT_NAME', body: buffer};
// Call Glacier to upload the archive.
glacier.uploadArchive(params, function(err, data) {
  if (err) {
    console.log("Error uploading archive!", err);
  } else {
    console.log("Archive ID", data.archiveId);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [UploadArchive](#) in [AWS SDK for JavaScript API Reference](#).

## Amazon SES examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon Simple Email Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3114\)](#)

## Actions

### List identities

The following code example shows how to list Amazon SES identities.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import { ListIdentitiesCommand } from "@aws-sdk/client-ses";
```

```
import { sesClient } from "./libs/sesClient.js";

const createListIdentitiesCommand = () =>
  new ListIdentitiesCommand({ IdentityType: "EmailAddress", MaxItems: 10 });

const run = async () => {
  const listIdentitiesCommand = createListIdentitiesCommand();

  try {
    return await sesClient.send(listIdentitiesCommand);
  } catch (err) {
    console.log("Failed to list identities.", err);
    return err;
  }
};
```

- For API details, see [ListIdentities](#) in *AWS SDK for JavaScript API Reference*.

## Amazon SNS examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon Simple Notification Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3115\)](#)

## Actions

### Check whether a phone number is opted out

The following code example shows how to check whether a phone number is opted out of receiving Amazon SNS messages.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {CheckIfPhoneNumberIsOptedOutCommand } from "@aws-sdk/client-sns";
```

```
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const params = { phoneNumber: "353861230764" }; //PHONE_NUMBER, in the E.164 phone
number structure

const run = async () => {
  try {
    const data = await snsClient.send(
      new CheckIfPhoneNumberIsOptedOutCommand(params)
    );
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CheckIfPhoneNumberIsOptedOut](#) in [AWS SDK for JavaScript API Reference](#).

### Confirm an endpoint owner wants to receive messages

The following code example shows how to confirm the owner of an endpoint wants to receive Amazon SNS messages by validating the token sent to the endpoint by an earlier Subscribe action.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {ConfirmSubscriptionCommand } from "@aws-sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const params = {
  Token: "TOKEN", // Required. Token sent to the endpoint by an earlier Subscribe
  action. */
  TopicArn: "TOPIC_ARN", // Required
  AuthenticateOnUnsubscribe: "true", // 'true' or 'false'
};

const run = async () => {
  try {
    const data = await snsClient.send(new ConfirmSubscriptionCommand(params));
```

```
        console.log("Success.", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err.stack);
    }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ConfirmSubscription](#) in [AWS SDK for JavaScript API Reference](#).

## Create a topic

The following code example shows how to create an Amazon SNS topic.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {CreateTopicCommand} from "@aws-sdk/client-sns";
import {snsClient} from "./libs/snsClient.js";

// Set the parameters
const params = { Name: "TOPIC_NAME" }; //TOPIC_NAME

const run = async () => {
    try {
        const data = await snsClient.send(new CreateTopicCommand(params));
        console.log("Success.", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err.stack);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateTopic](#) in [AWS SDK for JavaScript API Reference](#).

## Delete a subscription

The following code example shows how to delete an Amazon SNS subscription.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {UnsubscribeCommand } from "@aws-sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const params = { SubscriptionArn: "TOPIC_SUBSCRIPTION_ARN" }; //TOPIC_SUBSCRIPTION_ARN

const run = async () => {
  try {
    const data = await snsClient.send(new UnsubscribeCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Unsubscribe](#) in [AWS SDK for JavaScript API Reference](#).

## Delete a topic

The following code example shows how to delete an Amazon SNS topic and all subscriptions to that topic.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Load the AWS SDK for Node.js

// Import required AWS SDK clients and commands for Node.js
import {DeleteTopicCommand} from "@aws-sdk/client-sns";
import {snsClient} from "./libs/snsClient.js";

// Set the parameters
const params = { TopicArn: "TOPIC_ARN" }; //TOPIC_ARN

const run = async () => {
  try {
    const data = await snsClient.send(new DeleteTopicCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteTopic](#) in [AWS SDK for JavaScript API Reference](#).

## Get the properties of a topic

The following code example shows how to get the properties of an Amazon SNS topic.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {GetTopicAttributesCommand} from "@aws-sdk/client-sns";
import {snsClient} from "./libs/snsClient.js";

// Set the parameters
const params = { TopicArn: "TOPIC_ARN" }; // TOPIC_ARN

const run = async () => {
  try {
    const data = await snsClient.send(new GetTopicAttributesCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.error("Error", err.stack);
  }
};
run();
```

```
    } catch (err) {
      console.log("Error", err.stack);
    }
  };
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetTopicAttributes](#) in [AWS SDK for JavaScript API Reference](#).

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Import the SDK and client modules and call the API.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set region
AWS.config.update({region: 'REGION'});

// Create promise and SNS service object
var getTopicAttrbsPromise = new AWS.SNS({apiVersion:
  '2010-03-31'}).getTopicAttributes({TopicArn: 'TOPIC_ARN'}).promise();

// Handle promise's fulfilled/rejected states
getTopicAttrbsPromise.then(
  function(data) {
    console.log(data);
  }).catch(
  function(err) {
    console.error(err, err.stack);
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetTopicAttributes](#) in [AWS SDK for JavaScript API Reference](#).

### Get the settings for sending SMS messages

The following code example shows how to get the settings for sending Amazon SNS SMS messages.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {GetSMSAttributesCommand } from "@aws-sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
var params = {
  attributes: [
    "DefaultSMSType",
    "ATTRIBUTE_NAME",
    /* more items */
  ],
};

const run = async () => {
  try {
    const data = await snsClient.send(new GetSMSAttributesCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetSMSAttributes in AWS SDK for JavaScript API Reference](#).

## List the subscribers of a topic

The following code example shows how to retrieve the list of subscribers of an Amazon SNS topic.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {ListSubscriptionsByTopicCommand } from "@aws-sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const params = { TopicArn: "TOPIC_ARN" }; //TOPIC_ARN

const run = async () => {
```

```
try {
  const data = await snsClient.send(new ListSubscriptionsByTopicCommand(params));
  console.log("Success.", data);
  return data; // For unit tests.
} catch (err) {
  console.log("Error", err.stack);
}
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListSubscriptions](#) in [AWS SDK for JavaScript API Reference](#).

## List topics

The following code example shows how to list Amazon SNS topics.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {ListTopicsCommand} from "@aws-sdk/client-sns";
import {snsClient} from "./libs/snsClient.js";

const run = async () => {
  try {
    const data = await snsClient.send(new ListTopicsCommand([]));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListTopics](#) in [AWS SDK for JavaScript API Reference](#).

## Publish to a topic

The following code example shows how to publish messages to an Amazon SNS topic.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {PublishCommand} from "@aws-sdk/client-sns";
import {snsClient} from "./libs/snsClient.js";

// Set the parameters
var params = {
  Message: "MESSAGE_TEXT", // MESSAGE_TEXT
  TopicArn: "TOPIC_ARN", //TOPIC_ARN
};

const run = async () => {
  try {
    const data = await snsClient.send(new PublishCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Publish](#) in [AWS SDK for JavaScript API Reference](#).

## Set the default settings for sending SMS messages

The following code example shows how to set the default settings for sending SMS messages using Amazon SNS.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
```

```
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {SetSMSAttributesCommand} from "@aws-sdk/client-sns";
import {snsClient} from "./libs/snsClient.js";

// Set the parameters
const params = {
  attributes: {
    /* required */
    DefaultSMSType: "Transactional" /* highest reliability */,
    //DefaultSMSType: 'Promotional' /* lowest cost */
  },
};

const run = async () => {
  try {
    const data = await snsClient.send(new SetSMSAttributesCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SetSmsAttributes](#) in [AWS SDK for JavaScript API Reference](#).

## Set topic attributes

The following code example shows how to set Amazon SNS topic attributes.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {SetTopicAttributesCommand} from "@aws-sdk/client-sns";
```

```
import {snsClient} from "./libs/snsClient.js";

// Set the parameters
const params = {
  AttributeName: "ATTRIBUTE_NAME", // ATTRIBUTE_NAME
  TopicArn: "TOPIC_ARN", // TOPIC_ARN
  AttributeValue: "NEW_ATTRIBUTE_VALUE", //NEW_ATTRIBUTE_VALUE
};

const run = async () => {
  try {
    const data = await snsClient.send(new SetTopicAttributesCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SetTopicAttributes](#) in [AWS SDK for JavaScript API Reference](#).

### Subscribe a Lambda function to a topic

The following code example shows how to subscribe a Lambda function so it receives notifications from an Amazon SNS topic.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {SubscribeCommand} from "@aws-sdk/client-sns";
import {snsClient} from "./libs/snsClient.js";

// Set the parameters
const params = {
  Protocol: "lambda" /* required */,
  TopicArn: "TOPIC_ARN", //TOPIC_ARN
  Endpoint: "LAMBDA_FUNCTION_ARN", //LAMBDA_FUNCTION_ARN
};

const run = async () => {
  try {
    const data = await snsClient.send(new SubscribeCommand(params));
```

```
        console.log("Success.", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err.stack);
    }
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Subscribe](#) in [AWS SDK for JavaScript API Reference](#).

## Subscribe a mobile application to a topic

The following code example shows how to subscribe a mobile application endpoint so it receives notifications from an Amazon SNS topic.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {SubscribeCommand} from "@aws-sdk/client-sns";
import {snsClient} from "./libs/snsClient.js";

// Set the parameters
const params = {
    Protocol: "application" /* required */,
    TopicArn: "TOPIC_ARN", //TOPIC_ARN
    Endpoint: "MOBILE_ENDPOINT_ARN", // MOBILE_ENDPOINT_ARN
};

const run = async () => {
    try {
        const data = await snsClient.send(new SubscribeCommand(params));
        console.log("Success.", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err.stack);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Subscribe](#) in [AWS SDK for JavaScript API Reference](#).

## Subscribe an email address to a topic

The following code example shows how to subscribe an email address to an Amazon SNS topic.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client in a separate module and export it.

```
import { SNSClient } from "@aws-sdk/client-sns";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SNS service object.
const snsClient = new SNSClient({ region: REGION });
export { snsClient };
```

Import the SDK and client modules and call the API.

```
// Import required AWS SDK clients and commands for Node.js
import {SubscribeCommand } from "@aws-sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const params = {
  Protocol: "email" /* required */,
  TopicArn: "TOPIC_ARN", //TOPIC_ARN
  Endpoint: "EMAIL_ADDRESS", //EMAIL_ADDRESS
};

const run = async () => {
  try {
    const data = await snsClient.send(new SubscribeCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [Subscribe](#) in [AWS SDK for JavaScript API Reference](#).

## Amazon SQS examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon Simple Queue Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3128\)](#)

## Actions

### Change how long a queue waits for a message

The following code example shows how to change how long an Amazon SQS queue waits for a message to arrive.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Change how long an Amazon SQS queue waits for a message to arrive.

```
// Import required AWS SDK clients and commands for Node.js
import { SetQueueAttributesCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = {
  Attributes: {
    ReceiveMessageWaitTimeSeconds: "20",
  },
  QueueUrl: "SQS_QUEUE_URL", //SQS_QUEUE_URL; e.g., 'https://sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'
};

const run = async () => {
  try {
    const data = await sqsClient.send(new SetQueueAttributesCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.error(err, err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SetQueueAttributes](#) in [AWS SDK for JavaScript API Reference](#).

#### SDK for JavaScript V2

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Change how long an Amazon SQS queue waits for a message to arrive.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {
  Attributes: {
    "ReceiveMessageWaitTimeSeconds": "20",
  },
  QueueUrl: "SQS_QUEUE_URL"
};

sqs.setQueueAttributes(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SetQueueAttributes](#) in [AWS SDK for JavaScript API Reference](#).

### Change message timeout visibility

The following code example shows how to change an Amazon SQS message timeout visibility.

#### SDK for JavaScript V3

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Receive an Amazon SQS message and change its timeout visibility.

```
// Import required AWS SDK clients and commands for Node.js
import {
  ReceiveMessageCommand,
  ChangeMessageVisibilityCommand,
} from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const queueURL = "https://sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME"; // REGION,
  ACCOUNT_ID, QUEUE_NAME
const params = {
  AttributeNames: ["SentTimestamp"],
```

```
    MaxNumberOfMessages: 1,
    MessageAttributeNames: ["All"],
    QueueUrl: queueURL,
};

const run = async () => {
  try {
    const data = await sqsClient.send(new ReceiveMessageCommand(params));
    if (data.Messages != null) {
      try {
        const visibilityParams = {
          QueueUrl: queueURL,
          ReceiptHandle: data.Messages[0].ReceiptHandle,
          VisibilityTimeout: 20, // 20 second timeout
        };
        const results = await sqsClient.send(
          new ChangeMessageVisibilityCommand(visibilityParams)
        );
        console.log("Timeout Changed", results);
      } catch (err) {
        console.log("Delete Error", err);
      }
    } else {
      console.log("No messages to change");
    }
    return data; // For unit tests.
  } catch (err) {
    console.log("Receive Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ChangeMessageVisibility](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Receive an Amazon SQS message and change its timeout visibility.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk')
// Set the region to us-west-2
AWS.config.update({ region: 'us-west-2' })

// Create the SQS service object
var sqs = new AWS.SQS({ apiVersion: '2012-11-05' })

var queueURL = 'https://sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'

var params = {
  AttributeNames: ['SentTimestamp'],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ['All'],
  QueueUrl: queueURL
}

sqs.receiveMessage(params, function (err, data) {
  if (err) {
    console.log('Receive Error', err)
```

```
    } else {
        // Make sure we have a message
        if (data.Messages != null) {
            var visibilityParams = {
                QueueUrl: queueURL,
                ReceiptHandle: data.Messages[0].ReceiptHandle,
                VisibilityTimeout: 20 // 20 second timeout
            }
            sqs.changeMessageVisibility(visibilityParams, function (err, data) {
                if (err) {
                    console.log('Delete Error', err)
                } else {
                    console.log('Timeout Changed', data)
                }
            })
        } else {
            console.log('No messages to change')
        }
    })
})
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ChangeMessageVisibility](#) in [AWS SDK for JavaScript API Reference](#).

## Create a queue

The following code example shows how to create an Amazon SQS queue.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Create an Amazon SQS standard queue.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateQueueCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = {
    QueueName: "SQS_QUEUE_NAME", //SQS_QUEUE_URL
    Attributes: {
        DelaySeconds: "60", // Number of seconds delay.
        MessageRetentionPeriod: "86400", // Number of seconds delay.
    },
};

const run = async () => {
    try {
```

```
    const data = await sqsClient.send(new CreateQueueCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
} catch (err) {
    console.log("Error", err);
}
};

run();
```

Create an Amazon SQS queue that waits for a message to arrive.

```
// Import required AWS SDK clients and commands for Node.js
import { CreateQueueCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = {
    QueueName: "SQS_QUEUE_NAME", //SQS_QUEUE_URL; e.g., 'https://
sqs.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'
    Attributes: {
        ReceiveMessageWaitTimeSeconds: "20",
    },
};

const run = async () => {
    try {
        const data = await sqsClient.send(new CreateQueueCommand(params));
        console.log("Success", data);
        return data; // For unit tests.
    } catch (err) {
        console.error(err, err.stack);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateQueue in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an Amazon SQS standard queue.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {
    QueueName: 'SQS_QUEUE_NAME',
    Attributes: {
        'DelaySeconds': '60',
        'MessageRetentionPeriod': '86400'
    }
};
```

```
    sqs.createQueue(params, function(err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data.QueueUrl);
      }
});
```

Create an Amazon SQS queue that waits for a message to arrive.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {
  QueueName: 'SQS_QUEUE_NAME',
  Attributes: {
    'ReceiveMessageWaitTimeSeconds': '20',
  }
};

sqs.createQueue(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [CreateQueue](#) in [AWS SDK for JavaScript API Reference](#).

## Delete a message from a queue

The following code example shows how to delete a message from an Amazon SQS queue.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Receive and delete Amazon SQS messages.

```
// Import required AWS SDK clients and commands for Node.js
import {
  ReceiveMessageCommand,
  DeleteMessageCommand,
} from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const queueURL = "SQS_QUEUE_URL"; //SQS_QUEUE_URL; e.g., 'https://
sq.s.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'
const params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 10,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  VisibilityTimeout: 20,
  WaitTimeSeconds: 0,
};

const run = async () => {
  try {
    const data = await sqsClient.send(new ReceiveMessageCommand(params));
    if (data.Messages) {
      var deleteParams = {
        QueueUrl: queueURL,
        ReceiptHandle: data.Messages[0].ReceiptHandle,
      };
      try {
        const data = await sqsClient.send(new DeleteMessageCommand(deleteParams));
        console.log("Message deleted", data);
      } catch (err) {
        console.log("Error", err);
      }
    } else {
      console.log("No messages to delete");
    }
    return data; // For unit tests.
  } catch (err) {
    console.log("Receive Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteMessage](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Receive and delete Amazon SQS messages.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var queueURL = "SQS_QUEUE_URL";
```

```
var params = {
  AttributeNames: [
    "SentTimestamp"
  ],
  MaxNumberOfMessages: 10,
  MessageAttributeNames: [
    "All"
  ],
  QueueUrl: queueURL,
  VisibilityTimeout: 20,
  WaitTimeSeconds: 0
};

sqS.receiveMessage(params, function(err, data) {
  if (err) {
    console.log("Receive Error", err);
  } else if (data.Messages) {
    var deleteParams = {
      QueueUrl: queueURL,
      ReceiptHandle: data.Messages[0].ReceiptHandle
    };
    sqS.deleteMessage(deleteParams, function(err, data) {
      if (err) {
        console.log("Delete Error", err);
      } else {
        console.log("Message Deleted", data);
      }
    });
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteMessage](#) in [AWS SDK for JavaScript API Reference](#).

## Delete a queue

The following code example shows how to delete an Amazon SQS queue.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Delete an Amazon SQS queue.

```
// Import required AWS SDK clients and commands for Node.js
import { DeleteQueueCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";
```

```
// Set the parameters
const params = { QueueUrl: "SQS_QUEUE_URL" }; //SQS_QUEUE_URL e.g., 'https://
sq.s.REGION.amazonaws.com/ACCOUNT-ID/QUEUE-NAME'

const run = async () => {
  try {
    const data = await sqsClient.send(new DeleteQueueCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.error(err, err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteQueue](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an Amazon SQS queue.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {
  QueueUrl: 'SQS_QUEUE_URL'
};

sqs.deleteQueue(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteQueue](#) in [AWS SDK for JavaScript API Reference](#).

## Get the URL of a queue

The following code example shows how to get the URL of an Amazon SQS queue.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Get the URL for an Amazon SQS queue.

```
// Import required AWS SDK clients and commands for Node.js
import { GetQueueUrlCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = { QueueName: "SQS_QUEUE_NAME" };

const run = async () => {
  try {
    const data = await sqsClient.send(new GetQueueUrlCommand(params));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [GetQueueUrl](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get the URL for an Amazon SQS queue.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {
  QueueName: 'SQS_QUEUE_NAME'
};

sqs.getQueueUrl(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.QueueUrl);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [GetQueueUrl](#) in *AWS SDK for JavaScript API Reference*.

## List queues

The following code example shows how to list Amazon SQS queues.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

List your Amazon SQS queues.

```
// Import required AWS SDK clients and commands for Node.js
import { ListQueuesCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

const run = async () => {
  try {
    const data = await sqsClient.send(new ListQueuesCommand({}));
    console.log("Success", data);
    return data; // For unit tests.
  } catch (err) {
    console.error(err, err.stack);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListQueues](#) in *AWS SDK for JavaScript API Reference*.

### SDK for JavaScript V2

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List your Amazon SQS queues.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var params = {};
sqs.listQueues(params, function(err, data) {
```

```
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data.QueueUrls);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListQueues](#) in *AWS SDK for JavaScript API Reference*.

## Receive messages from a queue

The following code example shows how to receive messages from an Amazon SQS queue.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

## Receive a message from an Amazon SQS queue using long-poll support.

```
// Import required AWS SDK clients and commands for Node.js
import { ReceiveMessageCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const queueURL = "SQS_QUEUE_URL"; // SQS_QUEUE_URL
const params = {
  AttributeNames: ["SentTimestamp"],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: ["All"],
  QueueUrl: queueURL,
  WaitTimeSeconds: 20,
};

const run = async () => {
  try {
    const data = await sqsClient.send(new ReceiveMessageCommand(params));
    console.log("Success, ", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).

- For API details, see [ReceiveMessage in AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Receive a message from an Amazon SQS queue using long-poll support.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});

var queueURL = "SQS_QUEUE_URL";

var params = {
  AttributeNames: [
    "SentTimestamp"
  ],
  MaxNumberOfMessages: 1,
  MessageAttributeNames: [
    "All"
  ],
  QueueUrl: queueURL,
  WaitTimeSeconds: 20
};

sqs.receiveMessage(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ReceiveMessage in AWS SDK for JavaScript API Reference](#).

## Send a message to a queue

The following code example shows how to send a message to an Amazon SQS queue.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { SQSClient } from "@aws-sdk/client-sqs";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create SQS service object.
const sqsClient = new SQSClient({ region: REGION });
export { sqsClient };
```

Send a message to an Amazon SQS queue.

```
// Import required AWS SDK clients and commands for Node.js
import { SendMessageCommand } from "@aws-sdk/client-sqs";
import { sqsClient } from "./libs/sqsClient.js";

// Set the parameters
const params = {
  DelaySeconds: 10,
  MessageAttributes: {
    Title: {
      DataType: "String",
      StringValue: "The Whistler",
    },
    Author: {
      DataType: "String",
      StringValue: "John Grisham",
    },
    WeeksOn: {
      DataType: "Number",
      StringValue: "6",
    },
  },
  MessageBody:
    "Information about current NY Times fiction bestseller for week of 12/11/2016.",
  // MessageDeduplicationId: "TheWhistler", // Required for FIFO queues
  // MessageGroupId: "Group1", // Required for FIFO queues
  QueueUrl: "SQS_QUEUE_URL" //SQS_QUEUE_URL; e.g., 'https://sqs.REGION.amazonaws.com/
ACCOUNT-ID/QUEUE-NAME'
};

const run = async () => {
  try {
    const data = await sqsClient.send(new SendMessageCommand(params));
    console.log("Success, message sent. MessageID:", data.MessageId);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SendMessage](#) in [AWS SDK for JavaScript API Reference](#).

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Send a message to an Amazon SQS queue.

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create an SQS service object
var sqs = new AWS.SQS({apiVersion: '2012-11-05'});
```

```
var params = {
    // Remove DelaySeconds parameter and value for FIFO queues
    DelaySeconds: 10,
    MessageAttributes: {
        "Title": {
            DataType: "String",
            StringValue: "The Whistler"
        },
        "Author": {
            DataType: "String",
            StringValue: "John Grisham"
        },
        "WeeksOn": {
            DataType: "Number",
            StringValue: "6"
        }
    },
    MessageBody: "Information about current NY Times fiction bestseller for week of
12/11/2016.",
    // MessageDeduplicationId: "TheWhistler", // Required for FIFO queues
    // MessageGroupId: "Group1", // Required for FIFO queues
    QueueUrl: "SQS_QUEUE_URL"
};

sqS.sendMessage(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data.MessageId);
    }
});
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [SendMessage](#) in [AWS SDK for JavaScript API Reference](#).

## AWS STS examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with AWS Security Token Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3142\)](#)

## Actions

### Assume a role

The following code example shows how to assume a role with AWS STS.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon STS service client object.
const stsClient = new STSClient({ region: REGION });
export { stsClient };
```

Assume the IAM role.

```
// Import required AWS SDK clients and commands for Node.js
import { stsClient } from "./libs/stsClient.js";
import {
    AssumeRoleCommand,
    GetCallerIdentityCommand,
} from "@aws-sdk/client-sts";

// Set the parameters
export const params = {
    RoleArn: "ARN_OF_ROLE_TO_ASSUME", //ARN_OF_ROLE_TO_ASSUME
    RoleSessionName: "session1",
    DurationSeconds: 900,
};

export const run = async () => {
    try {
        //Assume Role
        const data = await stsClient.send(new AssumeRoleCommand(params));
        return data;
        const rolecreds = {
            accessKeyId: data.Credentials.AccessKeyId,
            secretAccessKey: data.Credentials.SecretAccessKey,
            sessionToken: data.Credentials.SessionToken,
        };
        //Get Amazon Resource Name (ARN) of current identity
        try {
            const stsParams = { credentials: rolecreds };
            const stsClient = new STSClient(stsParams);
            const results = await stsClient.send(
                new GetCallerIdentityCommand(rolecreds)
            );
            console.log("Success", results);
        } catch (err) {
            console.log(err, err.stack);
        }
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For API details, see [AssumeRole](#) in *AWS SDK for JavaScript API Reference*.

## SDK for JavaScript V2

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Load the AWS SDK for Node.js
```

```
const AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

var roleToAssume = {RoleArn: 'arn:aws:iam::123456789012:role/RoleName',
                    RoleSessionName: 'session1',
                    DurationSeconds: 900,};
var roleCreds;

// Create the STS service object
var sts = new AWS.STS({apiVersion: '2011-06-15'});

//Assume Role
sts.assumeRole(roleToAssume, function(err, data) {
    if (err) console.log(err, err.stack);
    else{
        roleCreds = {accessKeyId: data.Credentials.AccessKeyId,
                     secretAccessKey: data.Credentials.SecretAccessKey,
                     sessionToken: data.Credentials.SessionToken};
        stsGetCallerIdentity(roleCreds);
    }
});

//Get Arn of current identity
function stsGetCallerIdentity(creds) {
    var stsParams = {credentials: creds };
    // Create STS service object
    var sts = new AWS.STS(stsParams);

    sts.getCallerIdentity({}, function(err, data) {
        if (err) {
            console.log(err, err.stack);
        }
        else {
            console.log(data.ArN);
        }
    });
}
```

- For API details, see [AssumeRole](#) in *AWS SDK for JavaScript API Reference*.

## Amazon Transcribe examples using SDK for JavaScript V3

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for JavaScript V3 with Amazon Transcribe.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3144\)](#)

## Actions

### Delete a medical transcription job

The following code example shows how to delete an Amazon Transcribe Medical transcription job.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Delete a medical transcription job.

```
// Import the required AWS SDK clients and commands for Node.js
import { DeleteMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // For example,
  'medical_transcription_demo'
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new DeleteMedicalTranscriptionJobCommand(params)
    );
    console.log("Success - deleted");
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteMedicalTranscriptionJob](#) in [AWS SDK for JavaScript API Reference](#).

## Delete a transcription job

The following code example shows how to delete an Amazon Transcribe transcription job.

## SDK for JavaScript V3

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
```

```
// Create an Amazon Transcribe service client object.  
const transcribeClient = new TranscribeClient({ region: REGION });  
export { transcribeClient };
```

Delete a transcription job.

```
// Import the required AWS SDK clients and commands for Node.js  
import { DeleteTranscriptionJobCommand } from "@aws-sdk/client-transcribe";  
import { transcribeClient } from "./libs/transcribeClient.js";  
  
// Set the parameters  
export const params = {  
    TranscriptionJobName: "JOB_NAME", // Required. For example, 'transcription_demo'  
};  
  
export const run = async () => {  
    try {  
        const data = await transcribeClient.send(  
            new DeleteTranscriptionJobCommand(params)  
        );  
        console.log("Success - deleted");  
        return data; // For unit tests.  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [DeleteTranscriptionJob](#) in [AWS SDK for JavaScript API Reference](#).

## List medical transcription jobs

The following code example shows how to list Amazon Transcribe Medical transcription jobs.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");  
// Set the AWS Region.  
const REGION = "REGION"; //e.g. "us-east-1"  
// Create an Amazon Transcribe service client object.  
const transcribeClient = new TranscribeClient({ region: REGION });  
export { transcribeClient };
```

List medical transcription jobs.

```
// Import the required AWS SDK clients and commands for Node.js  
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";  
import { transcribeClient } from "./libs/transcribeClient.js";
```

```
// Set the parameters
export const params = {
    MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
    OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
    Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
    Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and 'DICTATION'
    LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
    MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
    Media: {
        MediaFileUri: "SOURCE_FILE_LOCATION",
        // The S3 object location of the input media file. The URI must be in the same
        // region
        // as the API endpoint that you are calling. For example,
        // "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
    },
};

export const run = async () => {
    try {
        const data = await transcribeClient.send(
            new StartMedicalTranscriptionJobCommand(params)
        );
        console.log("Success - put", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListMedicalTranscriptionJobs](#) in [AWS SDK for JavaScript API Reference](#).

## List transcription jobs

The following code example shows how to list Amazon Transcribe transcription jobs.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

List transcription jobs.

```
// Import the required AWS SDK clients and commands for Node.js
import { ListTranscriptionJobsCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";
```

```
// Set the parameters
export const params = {
  JobNameContains: "KEYWORD", // Not required. Returns only transcription
  // job names containing this string
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new ListTranscriptionJobsCommand(params)
    );
    console.log("Success", data.TranscriptionJobSummaries);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [ListTranscriptionJobs](#) in [AWS SDK for JavaScript API Reference](#).

## Start a medical transcription job

The following code example shows how to start an Amazon Transcribe Medical transcription job.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

## Start a medical transcription job.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartMedicalTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  MedicalTranscriptionJobName: "MEDICAL_JOB_NAME", // Required
  OutputBucketName: "OUTPUT_BUCKET_NAME", // Required
  Specialty: "PRIMARYCARE", // Required. Possible values are 'PRIMARYCARE'
  Type: "JOB_TYPE", // Required. Possible values are 'CONVERSATION' and 'DICTATION'
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_FILE_LOCATION",
    // The S3 object location of the input media file. The URI must be in the same
    region
    // as the API endpoint that you are calling. For example,
```

```
// "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
      new StartMedicalTranscriptionJobCommand(params)
    );
    console.log("Success - put", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [StartMedicalTranscriptionJob](#) in [AWS SDK for JavaScript API Reference](#).

## Start a transcription job

The following code example shows how to start an Amazon Transcribe transcription job.

### SDK for JavaScript V3

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the client.

```
const { TranscribeClient } = require("@aws-sdk/client-transcribe");
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon Transcribe service client object.
const transcribeClient = new TranscribeClient({ region: REGION });
export { transcribeClient };
```

Start a transcription job.

```
// Import the required AWS SDK clients and commands for Node.js
import { StartTranscriptionJobCommand } from "@aws-sdk/client-transcribe";
import { transcribeClient } from "./libs/transcribeClient.js";

// Set the parameters
export const params = {
  TranscriptionJobName: "JOB_NAME",
  LanguageCode: "LANGUAGE_CODE", // For example, 'en-US'
  MediaFormat: "SOURCE_FILE_FORMAT", // For example, 'wav'
  Media: {
    MediaFileUri: "SOURCE_LOCATION",
    // For example, "https://transcribe-demo.s3-REGION.amazonaws.com/hello_world.wav"
  },
  OutputBucketName: "OUTPUT_BUCKET_NAME"
};

export const run = async () => {
  try {
    const data = await transcribeClient.send(
```

```
    new StartTranscriptionJobCommand(params)
);
console.log("Success - put", data);
return data; // For unit tests.
} catch (err) {
  console.log("Error", err);
}
};

run();
```

- For more information, see [AWS SDK for JavaScript Developer Guide](#).
- For API details, see [StartTranscriptionJob](#) in [AWS SDK for JavaScript API Reference](#).

## Cross-service examples using SDK for JavaScript V3

The following sample applications use the AWS SDK for JavaScript V3 to work across multiple AWS services.

### Examples

- [Build an Amazon Transcribe app \(p. 3150\)](#)
- [Build an Amazon Transcribe streaming app \(p. 3151\)](#)
- [Build an application to submit data to a DynamoDB table \(p. 3151\)](#)
- [Create an Amazon Lex Chatbot within a web application to engage your web site visitors \(p. 3151\)](#)
- [Create a web application to track DynamoDB data \(p. 3152\)](#)
- [Create an Aurora Serverless work item tracker \(p. 3152\)](#)
- [Create an Amazon Textract explorer application \(p. 3152\)](#)
- [Detect PPE in images with Amazon Rekognition using an AWS SDK \(p. 3153\)](#)
- [Detect objects in images with Amazon Rekognition using an AWS SDK \(p. 3153\)](#)
- [Detect people and objects in a video with Amazon Rekognition using an AWS SDK \(p. 3154\)](#)
- [Invoke a Lambda function from a browser \(p. 3154\)](#)
- [Use API Gateway to invoke a Lambda function \(p. 3154\)](#)
- [Use Step Functions to invoke Lambda functions \(p. 3155\)](#)
- [Use scheduled events to invoke a Lambda function \(p. 3155\)](#)

## Build an Amazon Transcribe app

### SDK for JavaScript V3

Create an app that uses Amazon Transcribe to transcribe and display voice recordings in the browser. The app uses two Amazon Simple Storage Service (Amazon S3) buckets, one to host the application code, and another to store transcriptions. The app uses an Amazon Cognito user pool to authenticate your users. Authenticated users have AWS Identity and Access Management (IAM) permissions to access the required AWS services.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

### Services used in this example

- Amazon Cognito Identity

- Amazon S3
- Amazon Transcribe

## Build an Amazon Transcribe streaming app

### SDK for JavaScript V3

Shows how to use Amazon Transcribe to build an app that records, transcribes, and translates live audio in real-time, and emails the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Comprehend
- Amazon SES
- Amazon Transcribe
- Amazon Translate

## Build an application to submit data to a DynamoDB table

### SDK for JavaScript V3

This example shows how to build an app that enables users to submit data to an Amazon DynamoDB table, and send a text message to the administrator using Amazon Simple Notification Service (Amazon SNS).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

### Services used in this example

- DynamoDB
- Amazon SNS

## Create an Amazon Lex Chatbot within a web application to engage your web site visitors

### SDK for JavaScript V3

Shows how to use the Amazon Lex API to create a Chatbot within a web application to engage your web site visitors.

For complete source code and instructions on how to set up and run, see the full example [Building an Amazon Lex chatbot](#) in the AWS SDK for JavaScript developer guide.

### Services used in this example

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

## Create a web application to track DynamoDB data

### SDK for JavaScript V3

Shows how to use the Amazon DynamoDB API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon SES

## Create an Aurora Serverless work item tracker

### SDK for JavaScript V3

Shows how to use the AWS SDK for JavaScript (v3) to create a web application that tracks work items in an Amazon Aurora database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with an Express Node.js backend.

- Integrate a React.js web application with AWS services.
- List, add, and update items in an Aurora table.
- Send an email report of filtered work items by using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

## Create an Amazon Textract explorer application

### SDK for JavaScript V3

Shows how to use the AWS SDK for JavaScript to build a React application that uses Amazon Textract to extract data from a document image and display it in an interactive web page. This example runs in a web browser and requires an authenticated Amazon Cognito identity for credentials. It uses Amazon Simple Storage Service (Amazon S3) for storage, and for notifications it polls an Amazon Simple Queue Service (Amazon SQS) queue that is subscribed to an Amazon Simple Notification Service (Amazon SNS) topic.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS

- Amazon SQS
- Amazon Textract

## Detect PPE in images with Amazon Rekognition using an AWS SDK

### SDK for JavaScript V3

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an application to detect personal protective equipment (PPE) in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app saves the results to an Amazon DynamoDB table, and sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for PPE using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Update a DynamoDB table with results.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

## Detect objects in images with Amazon Rekognition using an AWS SDK

### SDK for JavaScript V3

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for objects using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Detect people and objects in a video with Amazon Rekognition using an AWS SDK

### SDK for JavaScript V3

Shows how to use Amazon Rekognition with the AWS SDK for JavaScript to create an app to detect faces and objects in videos located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

Learn how to:

- Create an unauthenticated user using Amazon Cognito.
- Analyze images for PPE using Amazon Rekognition.
- Verify an email address for Amazon SES.
- Send an email notification using Amazon SES.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Invoke a Lambda function from a browser

### SDK for JavaScript V2

You can create a browser-based application that uses an AWS Lambda function to update an Amazon DynamoDB table with user selections.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Lambda

### SDK for JavaScript V3

You can create a browser-based application that uses an AWS Lambda function to update an Amazon DynamoDB table with user selections. This app uses AWS SDK for JavaScript v3.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Lambda

## Use API Gateway to invoke a Lambda function

### SDK for JavaScript V3

Shows how to create an AWS Lambda function by using the Lambda JavaScript runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create a Lambda function invoked by Amazon API Gateway that scans an Amazon DynamoDB

table for work anniversaries and uses Amazon Simple Notification Service (Amazon SNS) to send a text message to your employees that congratulates them at their one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

#### Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

## Use Step Functions to invoke Lambda functions

### SDK for JavaScript V3

Shows how to create an AWS serverless workflow by using AWS Step Functions and the AWS SDK for JavaScript. Each workflow step is implemented using an AWS Lambda function.

Lambda is a compute service that enables you to run code without provisioning or managing servers. Step Functions is a serverless orchestration service that lets you combine Lambda functions and other AWS services to build business-critical applications.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

#### Services used in this example

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

## Use scheduled events to invoke a Lambda function

### SDK for JavaScript V3

Shows how to create an Amazon EventBridge scheduled event that invokes an AWS Lambda function. Configure EventBridge to use a cron expression to schedule when the Lambda function is invoked. In this example, you create a Lambda function by using the Lambda JavaScript runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create an app that sends a mobile text message to your employees that congratulates them at the one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

This example is also available in the [AWS SDK for JavaScript v3 developer guide](#).

#### Services used in this example

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

# Code examples for SDK for Java 2.x

The code examples in this topic show you how to use the AWS SDK for Java 2.x with AWS.

## Examples

- [Single-service actions and scenarios using SDK for Java 2.x \(p. 3156\)](#)
- [Cross-service examples using SDK for Java 2.x \(p. 3494\)](#)

## Single-service actions and scenarios using SDK for Java 2.x

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS services.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Services

- [API Gateway examples using SDK for Java 2.x \(p. 3157\)](#)
- [Application Recovery Controller examples using SDK for Java 2.x \(p. 3159\)](#)
- [Aurora examples using SDK for Java 2.x \(p. 3161\)](#)
- [CloudFront examples using SDK for Java 2.x \(p. 3182\)](#)
- [CloudWatch examples using SDK for Java 2.x \(p. 3184\)](#)
- [CloudWatch Events examples using SDK for Java 2.x \(p. 3189\)](#)
- [CloudWatch Logs examples using SDK for Java 2.x \(p. 3191\)](#)
- [Amazon Cognito Identity Provider examples using SDK for Java 2.x \(p. 3193\)](#)
- [Amazon Comprehend examples using SDK for Java 2.x \(p. 3205\)](#)
- [DynamoDB examples using SDK for Java 2.x \(p. 3209\)](#)
- [Amazon EC2 examples using SDK for Java 2.x \(p. 3238\)](#)
- [Amazon EC2 Auto Scaling examples using SDK for Java 2.x \(p. 3259\)](#)
- [EventBridge examples using SDK for Java 2.x \(p. 3273\)](#)
- [AWS Glue examples using SDK for Java 2.x \(p. 3275\)](#)
- [IAM examples using SDK for Java 2.x \(p. 3286\)](#)
- [AWS KMS examples using SDK for Java 2.x \(p. 3302\)](#)
- [Kinesis examples using SDK for Java 2.x \(p. 3308\)](#)
- [Lambda examples using SDK for Java 2.x \(p. 3312\)](#)
- [Amazon Personalize examples using SDK for Java 2.x \(p. 3319\)](#)
- [Amazon Personalize Events examples using SDK for Java 2.x \(p. 3338\)](#)
- [Amazon Personalize Runtime examples using SDK for Java 2.x \(p. 3340\)](#)
- [Amazon Pinpoint examples using SDK for Java 2.x \(p. 3343\)](#)
- [Amazon Pinpoint SMS and Voice API examples using SDK for Java 2.x \(p. 3358\)](#)
- [Amazon RDS examples using SDK for Java 2.x \(p. 3359\)](#)
- [Amazon Redshift examples using SDK for Java 2.x \(p. 3377\)](#)
- [Amazon Rekognition examples using SDK for Java 2.x \(p. 3379\)](#)

- [Amazon S3 examples using SDK for Java 2.x \(p. 3405\)](#)
- [S3 Glacier examples using SDK for Java 2.x \(p. 3432\)](#)
- [Amazon SES examples using SDK for Java 2.x \(p. 3439\)](#)
- [Amazon SES API v2 examples using SDK for Java 2.x \(p. 3444\)](#)
- [Amazon SNS examples using SDK for Java 2.x \(p. 3445\)](#)
- [Amazon SQS examples using SDK for Java 2.x \(p. 3461\)](#)
- [Secrets Manager examples using SDK for Java 2.x \(p. 3467\)](#)
- [Step Functions examples using SDK for Java 2.x \(p. 3471\)](#)
- [AWS Support examples using SDK for Java 2.x \(p. 3474\)](#)
- [Amazon Textract examples using SDK for Java 2.x \(p. 3490\)](#)

## API Gateway examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon API Gateway.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3157\)](#)

## Actions

### Create a REST API

The following code example shows how to create an API Gateway REST API.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createAPI( ApiGatewayClient apiGateway, String restApiId,
String restApiName) {

    try {
        CreateRestApiRequest request = CreateRestApiRequest.builder()
            .cloneFrom(restApiId)
            .description("Created using the Gateway Java API")
            .name(restApiName)
            .build();

        CreateRestApiResponse response = apiGateway.createRestApi(request);
        System.out.println("The id of the new api is "+response.id());
        return response.id();

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateRestApi](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a REST API

The following code example shows how to delete an API Gateway REST API.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteAPI( ApiGatewayClient apiGateway, String restApiId) {  
    try {  
        DeleteRestApiRequest request = DeleteRestApiRequest.builder()  
            .restApiId(restApiId)  
            .build();  
  
        apiGateway.deleteRestApi(request);  
        System.out.println("The API was successfully deleted");  
  
    } catch (ApiGatewayException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteRestApi](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a deployment

The following code example shows how to delete a deployment.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteSpecificDeployment(ApiGatewayClient apiGateway, String  
restApiId, String deploymentId) {  
  
    try {  
        DeleteDeploymentRequest request = DeleteDeploymentRequest.builder()  
            .restApiId(restApiId)  
            .deploymentId(deploymentId)  
            .build();  
  
        apiGateway.deleteDeployment(request);  
        System.out.println("Deployment was deleted" );  
  
    } catch (ApiGatewayException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

```
}
```

- For API details, see [DeleteDeployment](#) in *AWS SDK for Java 2.x API Reference*.

## Deploy a REST API

The following code example shows how to deploy an API Gateway REST API.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createNewDeployment(ApiGatewayClient apiGateway, String
restApiId, String stageName) {

    try {
        CreateDeploymentRequest request = CreateDeploymentRequest.builder()
            .restApiId(restApiId)
            .description("Created using the AWS API Gateway Java API")
            .stageName(stageName)
            .build();

        CreateDeploymentResponse response = apiGateway.createDeployment(request);
        System.out.println("The id of the deployment is "+response.id());
        return response.id();

    } catch (ApiGatewayException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateDeployment](#) in *AWS SDK for Java 2.x API Reference*.

## Application Recovery Controller examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Route 53 Application Recovery Controller.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3159\)](#)

## Actions

### Get the state of a routing control

The following code example shows how to get the state of an Application Recovery Controller routing control.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static GetRoutingControlStateResponse
getRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
                      String
routingControlArn) {
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region())).build();
            return client.getRoutingControlState(
                GetRoutingControlStateRequest.builder()
                    .routingControlArn(routingControlArn).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
    return null;
}
```

- For API details, see [GetRoutingControlState](#) in *AWS SDK for Java 2.x API Reference*.

## Update the state of a routing control

The following code example shows how to update the state of an Application Recovery Controller routing control.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static UpdateRoutingControlStateResponse
updateRoutingControlState(List<ClusterEndpoint> clusterEndpoints,
                          String
routingControlArn,
                          String
routingControlState) {
    for (ClusterEndpoint clusterEndpoint : clusterEndpoints) {
        try {
            System.out.println(clusterEndpoint);
            Route53RecoveryClusterClient client =
Route53RecoveryClusterClient.builder()
                .endpointOverride(URI.create(clusterEndpoint.endpoint()))
                .region(Region.of(clusterEndpoint.region()))
                .build();
            return client.updateRoutingControlState(
                UpdateRoutingControlStateRequest.builder()
                    .routingControlArn(routingControlArn).routingControlState(routingControlState).build());
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }
}
```

```
        }
    }
    return null;
}
```

- For API details, see [UpdateRoutingControlState](#) in *AWS SDK for Java 2.x API Reference*.

## Aurora examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Aurora.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3161\)](#)
- [Scenarios \(p. 3171\)](#)

## Actions

### Create a DB cluster

The following code example shows how to create an Aurora DB cluster.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName, String dbClusterIdentifier, String userName,
String password) {
    try {
        CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
            .databaseName(dbName)
            .dbClusterIdentifier(dbClusterIdentifier)
            .dbClusterParameterGroupName(dbParameterGroupFamily)
            .engine("aurora-mysql")
            .masterUsername(userName)
            .masterUserPassword(password)
            .build();

        CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
        return response.dbCluster().dbClusterArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateDBCluster](#) in *AWS SDK for Java 2.x API Reference*.

## Create a DB cluster parameter group

The following code example shows how to create an Aurora DB cluster parameter group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createDBClusterParameterGroup(RdsClient rdsClient, String dbClusterGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
            .dbClusterParameterGroupName(dbClusterGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateDBClusterParameterGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Create a DB cluster snapshot

The following code example shows how to create an Aurora DB cluster snapshot.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createDBClusterSnapshot(RdsClient rdsClient, String dbInstanceClusterIdentifier, String dbSnapshotIdentifier) {
    try {
        CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
        System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
    }
}
```

- For API details, see [CreateDBClusterSnapshot](#) in *AWS SDK for Java 2.x API Reference*.

## Create a DB instance in a DB cluster

The following code example shows how to create a DB instance in an Aurora DB cluster.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createDBInstanceCluster(RdsClient rdsClient,
                                             String dbInstanceIdentifier,
                                             String dbInstanceClusterIdentifier,
                                             String instanceClass){

    try {
        CreateDbInstanceRequest instanceRequest = CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbClusterIdentifier(dbInstanceClusterIdentifier)
            .engine("aurora-mysql")
            .dbInstanceClass(instanceClass)
            .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceState());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a DB cluster

The following code example shows how to delete an Aurora DB cluster.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
```

```
try {
    DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
    .dbClusterIdentifier(dbInstanceClusterIdentifier)
    .skipFinalSnapshot(true)
    .build();

    rdsClient.deleteDBCluster(deleteDbClusterRequest);
    System.out.println(dbInstanceClusterIdentifier +" was deleted!");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- For API details, see [DeleteDBCluster](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a DB cluster parameter group

The following code example shows how to delete an Aurora DB cluster parameter group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteDBClusterGroup( RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN) throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN ;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDBInstancesResponse response = rdsClient.describeDBInstances();
            List<DBInstance> instanceList = response.dbInstances();
            int listSize = instanceList.size();
            isDataDel = false ;
            didFind = false;
            int index = 1;
            for (DBInstance instance: instanceList) {
                instanceARN = instance.dbInstanceArn();
                if (instanceARN.compareTo(clusterDBARN) == 0) {
                    System.out.println(clusterDBARN + " still exists");
                    didFind = true ;
                }
                if ((index == listSize) && (!didFind)) {
                    // Went through the entire list and did not find the database
ARN.
                    isDataDel = true;
                }
                Thread.sleep(sleepTime * 1000);
                index++;
            }
        }

        DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest.builder()
    .dbClusterParameterGroupName(dbClusterGroupName)
```

```
        .build();

        rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
        System.out.println(dbClusterGroupName +" was deleted.");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteDBClusterParameterGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a DB instance

The following code example shows how to delete an Aurora DB instance.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteDatabaseInstance( RdsClient rdsClient, String
dbInstanceIdentifier ) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .deleteAutomatedBackups(true)
        .skipFinalSnapshot(true)
        .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

## Describe DB cluster parameter groups

The following code example shows how to describe Aurora DB cluster parameter groups.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String
dbClusterGroupName) {
```

```
try {
    DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
    .dbClusterParameterGroupName(dbClusterGroupName)
    .maxRecords(20)
    .build();

    List<DBClusterParameterGroup> groups =
rdsClient.describeDbClusterParameterGroups(groupsRequest).dbClusterParameterGroups();
    for (DBClusterParameterGroup group: groups) {
        System.out.println("The group name is
"+group.dbClusterParameterGroupName());
        System.out.println("The group ARN is
"+group.dbClusterParameterGroupArn());
    }

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- For API details, see [DescribeDBClusterParameterGroups](#) in *AWS SDK for Java 2.x API Reference*.

## Describe DB cluster snapshots

The following code example shows how to describe Aurora DB cluster snapshots.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier, String dbInstanceClusterIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");

        DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
    .dbClusterIdentifier(dbInstanceClusterIdentifier)
    .build();

        while (!snapshotReady) {
            DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
            List<DBClusterSnapshot> snapshotList = response.dbClusterSnapshots();
            for (DBClusterSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                } else {
                    System.out.println(".");
                    Thread.sleep(sleepTime * 5000);
                }
            }
        }
    }
}
```

```
        System.out.println("The Snapshot is available!");

    } catch (RdsException | InterruptedException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBClusterSnapshots](#) in *AWS SDK for Java 2.x API Reference*.

## Describe DB clusters

The following code example shows how to describe Aurora DB clusters.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDbClusterParameters(RdsClient rdsClient, String
dbClusterGroupName, int flag) {
    try {
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbClusterParametersRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .source("user")
                .build();
        }

        DescribeDbClusterParametersResponse response =
rdsClient.describeDBClusterParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para: dbParameters) {
            // Only print out information about either auto_increment_offset or
auto_increment_increment.
            paraName = para.parameterName();
            if ( (paraName.compareTo("auto_increment_offset") ==0) ||
(paraName.compareTo("auto_increment_increment ") ==0)) {
                System.out.println("**** The parameter name is " + paraName);
                System.out.println("**** The parameter value is " +
para.parameterValue());
                System.out.println("**** The parameter data type is " +
para.dataType());
                System.out.println("**** The parameter description is " +
para.description());
                System.out.println("**** The parameter allowed values is " +
para.allowedValues());
            }
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBClusters](#) in *AWS SDK for Java 2.x API Reference*.

## Describe DB instances

The following code example shows how to describe Aurora DB instances.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");

    try {
        DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
            .dbClusterIdentifier(dbClusterIdentifier)
            .build();

        while (!instanceReady) {
            DescribedDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
            List<DBCluster> clusterList = response.dbClusters();
            for (DBCluster cluster : clusterList) {
                instanceReadyStr = cluster.status();
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database cluster is available!");
    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Describe database engine versions

The following code example shows how to describe Aurora database engine versions.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDBEngines( RdsClient rdsClient ) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .engine("aurora-mysql")
    .defaultOnly(true)
    .maxRecords(20)
    .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engine0b: engines) {
            System.out.println("The name of the DB parameter group family for the
database engine is "+engine0b.dbParameterGroupFamily());
            System.out.println("The name of the database engine
"+engine0b.engine());
            System.out.println("The version number of the database engine
"+engine0b.engineVersion());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBEngineVersions in AWS SDK for Java 2.x API Reference](#).

## Describe options for DB instances

The following code example shows how to describe options for Aurora DB instances.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDBEngines( RdsClient rdsClient ) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .engine("aurora-mysql")
    .defaultOnly(true)
    .maxRecords(20)
    .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engine0b: engines) {
            System.out.println("The name of the DB parameter group family for the
database engine is "+engine0b.dbParameterGroupFamily());
            System.out.println("The name of the database engine
"+engine0b.engine());
```

```
        System.out.println("The version number of the database engine  
"+engine0b.engineVersion());  
    }  
  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DescribeOrderableDBInstanceOptions](#) in *AWS SDK for Java 2.x API Reference*.

## Describe parameters from a DB cluster parameter group

The following code example shows how to describe parameters from an Aurora DB cluster parameter group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDbClusterParameters(RdsClient rdsClient, String  
dbClusterGroupName, int flag) {  
    try {  
        DescribeDbClusterParametersRequest dbParameterGroupsRequest;  
        if (flag == 0) {  
            dbParameterGroupsRequest = DescribeDbClusterParametersRequest.builder()  
                .dbClusterParameterGroupName(dbClusterGroupName)  
                .build();  
        } else {  
            dbParameterGroupsRequest = DescribeDbClusterParametersRequest.builder()  
                .dbClusterParameterGroupName(dbClusterGroupName)  
                .source("user")  
                .build();  
        }  
  
        DescribeDbClusterParametersResponse response =  
rdsClient.describeDBClusterParameters(dbParameterGroupsRequest);  
        List<Parameter> dbParameters = response.parameters();  
        String paraName;  
        for (Parameter para: dbParameters) {  
            // Only print out information about either auto_increment_offset or  
auto_increment_increment.  
            paraName = para.parameterName();  
            if ( (paraName.compareTo("auto_increment_offset") ==0) ||  
(paraName.compareTo("auto_increment_increment ") ==0)) {  
                System.out.println("**** The parameter name is " + paraName);  
                System.out.println("**** The parameter value is " +  
para.parameterValue());  
                System.out.println("**** The parameter data type is " +  
para.dataType());  
                System.out.println("**** The parameter description is " +  
para.description());  
                System.out.println("**** The parameter allowed values is " +  
para.allowedValues());  
            }  
        }  
  
    } catch (RdsException e) {
```

```
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBClusterParameters](#) in *AWS SDK for Java 2.x API Reference*.

## Update parameters in a DB cluster parameter group

The following code example shows how to update parameters in an Aurora DB cluster parameter group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDbClusterParameterGroups(RdsClient rdsClient, String dbClusterGroupName) {
    try {
        DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
        .dbClusterParameterGroupName(dbClusterGroupName)
        .maxRecords(20)
        .build();

        List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest).dbClusterParameterGroups();
        for (DBClusterParameterGroup group: groups) {
            System.out.println("The group name is
"+group.dbClusterParameterGroupName());
            System.out.println("The group ARN is
"+group.dbClusterParameterGroupArn());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [ModifyDBClusterParameterGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Scenarios

### Get started with DB clusters

The following code example shows how to:

- Create a custom Aurora DB cluster parameter group and set parameter values.
- Create a DB cluster that is configured to use the parameter group.
- Create a DB instance in the DB cluster that contains a database.
- Take a snapshot of the DB cluster.
- Delete the instance, DB cluster, and parameter group.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
 * Before running this Java (v2) code example, set up your development environment,  
 * including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * This Java example performs the following tasks:  
 *  
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition by  
 * calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.  
 * 2. Selects an engine family and creates a custom DB cluster parameter group by  
 * invoking the describeDBClusterParameters method.  
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups  
 * method.  
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters method.  
 * 5. Modifies the auto_increment_offset parameter by invoking the  
 * modifyDBClusterParameterGroupRequest method.  
 * 6. Gets and displays the updated parameters.  
 * 7. Gets a list of allowed engine versions by invoking the describeDbEngineVersions  
 * method.  
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL database.  
 * 9. Waits for DB instance to be ready.  
 * 10. Gets a list of instance classes available for the selected engine.  
 * 11. Creates a database instance in the cluster.  
 * 12. Waits for DB instance to be ready.  
 * 13. Creates a snapshot.  
 * 14. Waits for DB snapshot to be ready.  
 * 15. Deletes the DB cluster.  
 * 16. Deletes the DB cluster group.  
 */  
  
public class AuroraScenario {  
    public static long sleepTime = 20;  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    public static void main(String[] args) throws InterruptedException {  
  
        final String usage = "\n" +  
            "Usage:\n" +  
            "      <dbClusterGroupName> <dbParameterGroupFamily>  
<dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName> <dbSnapshotIdentifier>  
<username> <userPassword>" +  
            "Where:\n" +  
            "      dbClusterGroupName - The name of the DB cluster parameter group. \n"+  
            "      dbParameterGroupFamily - The DB cluster parameter group family name  
(for example, aurora-mysql5.7). \n"+  
            "      dbInstanceClusterIdentifier - The instance cluster identifier value.  
\n"+  
            "      dbInstanceIdentifier - The database instance identifier.\n"+  
            "      dbName - The database name.\n"+  
            "      dbSnapshotIdentifier - The snapshot identifier.\n"+  
            "      username - The database user name.\n" +  
            "      userPassword - The database user name password.\n" ;  
  
        if (args.length != 8) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
    }  
}
```

```
}

String dbClusterGroupName = args[0];
String dbParameterGroupFamily = args[1];
String dbInstanceClusterIdentifier = args[2];
String dbInstanceIdentifier = args[3];
String dbName = args[4];
String dbSnapshotIdentifier = args[5];
String username = args[6];
String userPassword = args[7];

Region region = Region.US_WEST_2;
RdsClient rdsClient = RdsClient.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon Aurora example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Return a list of the available DB engines");
describeDBEngines(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a custom parameter group");
createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
    dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get the parameter group");
describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get the parameters in the group");
describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Modify the auto_increment_offset parameter");
modifyDBClusterParas(rdsClient, dbClusterGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Display the updated parameter value");
describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of allowed engine versions");
getAllowedEngines(rdsClient, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Create an Aurora DB cluster database");
String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName, dbName,
    dbInstanceClusterIdentifier, username, userPassword) ;
System.out.println("The ARN of the cluster is "+arnClusterVal);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Wait for DB instance to be ready" );
```

```
waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a list of instance classes available for the
selected engine");
String instanceClass = getListInstanceClasses(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Create a database instance in the cluster.");
String clusterDBARN = createDBInstanceCluster(rdsClient, dbInstanceIdentifier,
dbInstanceClusterIdentifier, instanceClass);
System.out.println("The ARN of the database is "+clusterDBARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Wait for DB instance to be ready" );
waitForDBInstanceReady(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Create a snapshot");
createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Wait for DB snapshot to be ready" );
waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Delete the DB instance" );
deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Delete the DB cluster");
deleteCluster(rdsClient, dbInstanceClusterIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Delete the DB cluster group");
deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The Scenario has successfully completed." );
System.out.println(DASHES);
rdsClient.close();
}

public static void deleteDBClusterGroup( RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN) throws InterruptedException {
try {
    boolean isDataDel = false;
    boolean didFind;
    String instanceARN ;

    // Make sure that the database has been deleted.
    while (!isDataDel) {
        DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
        List<DBInstance> instanceList = response.dbInstances();
        int listSize = instanceList.size();
        if (listSize > 0) {
            for (DBInstance instance : instanceList) {
                if (instance.dbClusterIdentifier.equals(dbClusterGroupName)) {
                    instanceARN = instance.arn;
                    isDataDel = true;
                }
            }
        } else {
            isDataDel = true;
        }
    }
}
}
```

```
        isDataDel = false ;
        didFind = false;
        int index = 1;
        for (DBInstance instance: instanceList) {
            instanceARN = instance.dbInstanceArn();
            if (instanceARN.compareTo(clusterDBARN) == 0) {
                System.out.println(clusterDBARN + " still exists");
                didFind = true ;
            }
            if ((index == listSize) && (!didFind)) {
                // Went through the entire list and did not find the database
                ARN.
                isDataDel = true;
            }
            Thread.sleep(sleepTime * 1000);
            index++;
        }
    }

    DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest.builder()
    .dbClusterParameterGroupName(dbClusterGroupName)
    .build();

    rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
    System.out.println(dbClusterGroupName +" was deleted.");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
    try {
        DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
    .dbClusterIdentifier(dbInstanceClusterIdentifier)
    .skipFinalSnapshot(true)
    .build();

        rdsClient.deleteDBCluster(deleteDbClusterRequest);
        System.out.println(dbInstanceClusterIdentifier +" was deleted!");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void deleteDatabaseInstance( RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
    .dbInstanceIdentifier(dbInstanceIdentifier)
    .deleteAutomatedBackups(true)
    .skipFinalSnapshot(true)
    .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.println("The status of the database is " +
response.dbInstance().dbInstanceState());
    }
}
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitForSnapshotReady(RdsClient rdsClient, String dbSnapshotIdentifier, String dbInstanceClusterIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                .dbClusterIdentifier(dbInstanceClusterIdentifier)
                .build();

            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList = response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }
            System.out.println("The Snapshot is available!");
        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterSnapshot(RdsClient rdsClient, String dbInstanceClusterIdentifier, String dbSnapshotIdentifier) {
        try {
            CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                .dbClusterIdentifier(dbInstanceClusterIdentifier)
                .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                .build();

            CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
            System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitDBInstanceReady(RdsClient rdsClient, String dbInstanceIdentifier) {
        boolean instanceReady = false;
```

```

        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");

        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .build();

            String endpoint="";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceState();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection endpoint
is "+ endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBInstanceCluster(RdsClient rdsClient,
                                                String dbInstanceIdentifier,
                                                String dbInstanceClusterIdentifier,
                                                String instanceClass){

        try {
            CreateDbInstanceRequest instanceRequest = CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .dbClusterIdentifier(dbInstanceClusterIdentifier)
                .engine("aurora-mysql")
                .dbInstanceClass(instanceClass)
                .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceState());
            return response.dbInstance().dBInstanceArn();

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String getListInstanceClasses(RdsClient rdsClient) {
        try{
            DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest.builder()
                .engine("aurora-mysql")

```

```
.maxRecords(20)
.build();

DescribeOrderableDbInstanceOptionsResponse response =
rdsClient.describeOrderableDBInstanceOptions(optionsRequest);
List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
String instanceClass = "";
for (OrderableDBInstanceOption instanceOption: instanceOptions) {
    instanceClass = instanceOption.dbInstanceClass();
    System.out.println("The instance class is "
+instanceOption.dbInstanceClass());
    System.out.println("The engine version is "
+instanceOption.engineVersion());
}
return instanceClass;

} catch (RdsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");

    try {
        DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
        .dbClusterIdentifier(dbClusterIdentifier)
        .build();

        while (!instanceReady) {
            DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
            List<DBCluster> clusterList = response.dbClusters();
            for (DBCluster cluster : clusterList) {
                instanceReadyStr = cluster.status();
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
        System.out.println("Database cluster is available!");
    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName, String dbClusterIdentifier, String userName,
String password) {
    try {
        CreateDbClusterRequest clusterRequest = CreateDbClusterRequest.builder()
        .databaseName(dbName)
```

```
.dbClusterIdentifier(dbClusterIdentifier)
.dbClusterParameterGroupName(dbParameterGroupFamily)
.engine("aurora-mysql")
.masterUsername(userName)
.masterUserPassword(password)
.build();

CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
return response.dbCluster().dbClusterArn();

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return "";
}

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
try {
    DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .dbParameterGroupFamily(dbParameterGroupFamily)
    .engine("aurora-mysql")
    .build();

    DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
    List<DBEngineVersion> dbEngines = response.dbEngineVersions();
    for (DBEngineVersion dbEngine: dbEngines) {
        System.out.println("The engine version is " +dbEngine.engineVersion());
        System.out.println("The engine description is "
+dbEngine.dbEngineDescription());
    }
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Modify the auto_increment_offset parameter.
public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
try {
    Parameter parameter1 = Parameter.builder()
        .parameterName("auto_increment_offset")
        .applyMethod("immediate")
        .parameterValue("5")
        .build();

    List<Parameter> paraList = new ArrayList<>();
    paraList.add(parameter1);
    ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
        .dbClusterParameterGroupName(dClusterGroupName)
        .parameters(paraList)
        .build();

    ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
    System.out.println("The parameter group "+
response.dbClusterParameterGroupName() +" was successfully modified");
}
```

```

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameters(RdsClient rdsClient, String dbClusterGroupName, int flag) {
        try {
            DescribeDbClusterParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest = DescribeDbClusterParametersRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();
            } else {
                dbParameterGroupsRequest = DescribeDbClusterParametersRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .source("user")
                    .build();
            }

            DescribeDbClusterParametersResponse response =
            rdsClient.describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para: dbParameters) {
                // Only print out information about either auto_increment_offset or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ( (paraName.compareTo("auto_increment_offset") ==0) ||
                (paraName.compareTo("auto_increment_increment ") ==0)) {
                    System.out.println("**** The parameter name is " + paraName);
                    System.out.println("**** The parameter value is " +
                    para.parameterValue());
                    System.out.println("**** The parameter data type is " +
                    para.dataType());
                    System.out.println("**** The parameter description is " +
                    para.description());
                    System.out.println("**** The parameter allowed values is " +
                    para.allowedValues());
                }
            }
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameterGroups(RdsClient rdsClient, String dbClusterGroupName) {
        try {
            DescribeDbClusterParameterGroupsRequest groupsRequest =
            DescribeDbClusterParameterGroupsRequest.builder()
                .dbClusterParameterGroupName(dbClusterGroupName)
                .maxRecords(20)
                .build();

            List<DBClusterParameterGroup> groups =
            rdsClient.describeDBClusterParameterGroups(groupsRequest).dbClusterParameterGroups();
            for (DBClusterParameterGroup group: groups) {
                System.out.println("The group name is
"+group.dbClusterParameterGroupName());
                System.out.println("The group ARN is
"+group.dbClusterParameterGroupArn());
        }
    }
}

```

```

        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void createDBClusterParameterGroup(RdsClient rdsClient, String dbClusterGroupName, String dbParameterGroupFamily) {
    try {
        CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
        .dbClusterParameterGroupName(dbClusterGroupName)
        .dbParameterGroupFamily(dbParameterGroupFamily)
        .description("Created by using the AWS SDK for Java")
        .build();

        CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
        System.out.println("The group name is "+
response.dbClusterParameterGroup().dbClusterParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDBEngines( RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
        .engine("aurora-mysql")
        .defaultOnly(true)
        .maxRecords(20)
        .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engine0b: engines) {
            System.out.println("The name of the DB parameter group family for the
database engine is "+engine0b.dbParameterGroupFamily());
            System.out.println("The name of the database engine
"+engine0b.engine());
            System.out.println("The version number of the database engine
"+engine0b.engineVersion());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}

```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.

- [CreateDBCluster](#)
- [CreateDBClusterParameterGroup](#)
- [CreateDBClusterSnapshot](#)

- [CreateDBInstance](#)
- [DeleteDBCluster](#)
- [DeleteDBClusterParameterGroup](#)
- [DeleteDBInstance](#)
- [DescribeDBClusterParameterGroups](#)
- [DescribeDBClusterParameters](#)
- [DescribeDBClusterSnapshots](#)
- [DescribeDBClusters](#)
- [DescribeDBEngineVersions](#)
- [DescribeDBInstances](#)
- [DescribeOrderableDBInstanceStateOptions](#)
- [ModifyDBClusterParameterGroup](#)

## CloudFront examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon CloudFront.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3182\)](#)

## Actions

### Create a function

The following code example shows how to create an Amazon CloudFront function.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createNewFunction(CloudFrontClient cloudFrontClient, String
functionName, String filePath) {

    try {

        InputStream is = new FileInputStream(filePath);
        SdkBytes functionCode = SdkBytes.fromInputStream(is);

        FunctionConfig config = FunctionConfig.builder()
            .comment("Created by using the CloudFront Java API")
            .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .name(functionName)
            .functionCode(functionCode)
            .functionConfig(config)
    }
}
```

```
        .build();

        CreateFunctionResponse response =
cloudFrontClient.createFunction(functionRequest);
        return response.functionSummary().functionMetadata().functionARN();

    } catch (CloudFrontException | FileNotFoundException e){
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateFunction](#) in *AWS SDK for Java 2.x API Reference*.

## Update a distribution

The following code example shows how to update an Amazon CloudFront distribution.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void modDistribution(CloudFrontClient cloudFrontClient, String idVal)
{
    try {
        // Get the Distribution to modify.
        GetDistributionRequest disRequest = GetDistributionRequest.builder()
            .id(idVal)
            .build();

        GetDistributionResponse response =
cloudFrontClient.getDistribution(disRequest);
        Distribution disObject = response.distribution();
        DistributionConfig config = disObject.distributionConfig();

        // Create a new DistributionConfig object and add new values to comment
and aliases
        DistributionConfig config1 = DistributionConfig.builder()
            .aliases(config.aliases()) // You can pass in new values here
            .comment("New Comment")
            .cacheBehaviors(config.cacheBehaviors())
            .priceClass(config.priceClass())
            .defaultCacheBehavior(config.defaultCacheBehavior())
            .enabled(config.enabled())
            .callerReference(config.callerReference())
            .logging(config.logging())
            .originGroups(config.originGroups())
            .origins(config.origins())
            .restrictions(config.restrictions())
            .defaultRootObject(config.defaultRootObject())
            .webACLId(config.webACLId())
            .httpVersion(config.httpVersion())
            .viewerCertificate(config.viewerCertificate())
            .customErrorResponses(config.customErrorResponses())
            .build();

        UpdateDistributionRequest updateDistributionRequest =
UpdateDistributionRequest.builder()
```

```
.distributionConfig(config1)
.id(disObject.id())
.ifMatch(response.eTag())
.build();

cloudFrontClient.updateDistribution(updateDistributionRequest);

} catch (CloudFrontException e){
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [UpdateDistribution](#) in *AWS SDK for Java 2.x API Reference*.

## CloudWatch examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon CloudWatch.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3184\)](#)

## Actions

### Create an alarm that watches a metric

The following code example shows how to create an Amazon CloudWatch alarm that watches a metric.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void putMetricAlarm(CloudWatchClient cw, String alarmName, String
instanceId) {

    try {
        Dimension dimension = Dimension.builder()
            .name("InstanceId")
            .value(instanceId).build();

        PutMetricAlarmRequest request = PutMetricAlarmRequest.builder()
            .alarmName(alarmName)
            .comparisonOperator(ComparisonOperator.GREATER_THAN_THRESHOLD)
            .evaluationPeriods(1)
            .metricName("CPUUtilization")
            .namespace("AWS/EC2")
            .period(60)
            .statistic(Statistic.AVERAGE)
            .threshold(70.0)
            .actionsEnabled(false)
            .alarmDescription("Alarm when server CPU utilization exceeds 70%")
    }
}
```

```
        .unit(StandardUnit.SECONDS)
        .dimensions(dimension)
        .build();

    cw.putMetricAlarm(request);
    System.out.printf("Successfully created alarm with name %s", alarmName);

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [PutMetricAlarm](#) in *AWS SDK for Java 2.x API Reference*.

## Delete alarms

The following code example shows how to delete Amazon CloudWatch alarms.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteCWAAlarm(CloudWatchClient cw, String alarmName) {

    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.printf("Successfully deleted alarm %s", alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteAlarms](#) in *AWS SDK for Java 2.x API Reference*.

## Describe alarms for a metric

The following code example shows how to describe Amazon CloudWatch alarms for a metric.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void desCWAAlarms( CloudWatchClient cw) {
    try {

        boolean done = false;
```

```
String newToken = null;

while(!done) {
    DescribeAlarmsResponse response;
    if (newToken == null) {
        DescribeAlarmsRequest request =
DescribeAlarmsRequest.builder().build();
        response = cw.describeAlarms(request);
    } else {
        DescribeAlarmsRequest request = DescribeAlarmsRequest.builder()
            .nextToken(newToken)
            .build();
        response = cw.describeAlarms(request);
    }

    for(MetricAlarm alarm : response.metricAlarms()) {
        System.out.printf("\n Retrieved alarm %s", alarm.alarmName());
    }

    if(response.nextToken() == null) {
        done = true;
    } else {
        newToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Done");
}
```

- For API details, see [DescribeAlarmsForMetric](#) in *AWS SDK for Java 2.x API Reference*.

## Disable alarm actions

The following code example shows how to disable Amazon CloudWatch alarm actions.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void disableActions(CloudWatchClient cw, String alarmName) {

    try {
        DisableAlarmActionsRequest request = DisableAlarmActionsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.disableAlarmActions(request);
        System.out.printf("Successfully disabled actions on alarm %s", alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DisableAlarmActions](#) in *AWS SDK for Java 2.x API Reference*.

## Enable alarm actions

The following code example shows how to enable Amazon CloudWatch alarm actions.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void enableActions(CloudWatchClient cw, String alarm) {  
  
    try {  
        EnableAlarmActionsRequest request = EnableAlarmActionsRequest.builder()  
            .alarmNames(alarm)  
            .build();  
  
        cw.enableAlarmActions(request);  
        System.out.printf("Successfully enabled actions on alarm %s", alarm);  
  
    } catch (CloudWatchException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for Java 2.x API Reference*.

## List metrics

The following code example shows how to list Amazon CloudWatch metrics.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listMets( CloudWatchClient cw, String namespace) {  
  
    boolean done = false;  
    String nextToken = null;  
  
    try {  
        while(!done) {  
  
            ListMetricsResponse response;  
            if (nextToken == null) {  
                ListMetricsRequest request = ListMetricsRequest.builder()  
                    .namespace(namespace)  
                    .build();  
  
                response = cw.listMetrics(request);  
            } else {  
                ListMetricsRequest request = ListMetricsRequest.builder()  
                    .namespace(namespace)  
                    .nextToken(nextToken)
```

```
        .build();

        response = cw.listMetrics(request);
    }

    for (Metric metric : response.metrics()) {
        System.out.printf("Retrieved metric %s", metric.metricName());
        System.out.println();
    }

    if(response.nextToken() == null) {
        done = true;
    } else {
        nextToken = response.nextToken();
    }
}

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [ListMetrics](#) in *AWS SDK for Java 2.x API Reference*.

## Put data into a metric

The following code example shows how to put data into a Amazon CloudWatch metric.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void putMetData(CloudWatchClient cw, Double dataPoint ) {

    try {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object.
        String time =
ZonedDateTime.now( ZoneOffset.UTC ).format( DateTimeFormatter.ISO_INSTANT );
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
            .value(dataPoint)
            .timestamp(instant)
            .dimensions(dimension).build();

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace("SITE/TRAFFIC")
            .metricData(datum).build();

        cw.putMetricData(request);

    } catch (CloudWatchException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.printf("Successfully put data point %f", dataPoint);
}
```

- For API details, see [PutMetricData](#) in *AWS SDK for Java 2.x API Reference*.

## CloudWatch Events examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon CloudWatch Events.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3189\)](#)

## Actions

### Adding a Lambda function target

The following code example shows how to add an AWS Lambda function target to an Amazon CloudWatch Events event.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void putCWTTargets(CloudWatchEventsClient cwe, String ruleName, String
functionArn, String targetId ) {

    try {
        Target target = Target.builder()
            .arn(functionArn)
            .id(targetId)
            .build();

        PutTargetsRequest request = PutTargetsRequest.builder()
            .targets(target)
            .rule(ruleName)
            .build();

        cwe.putTargets(request);
        System.out.printf(
            "Successfully created CloudWatch events target for rule %s",
            ruleName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutTargets](#) in *AWS SDK for Java 2.x API Reference*.

## Create a scheduled rule

The following code example shows how to create an Amazon CloudWatch Events scheduled rule.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void putCWRule(CloudWatchEventsClient cwe, String ruleName, String roleArn) {  
  
    try {  
        PutRuleRequest request = PutRuleRequest.builder()  
            .name(ruleName)  
            .roleArn(roleArn)  
            .scheduleExpression("rate(5 minutes)")  
            .state(RuleState.ENABLED)  
            .build();  
  
        PutRuleResponse response = cwe.putRule(request);  
        System.out.printf(  
            "Successfully created CloudWatch events rule %s with arn %s",  
            roleArn, response.ruleArn());  
  
    } catch (  
        CloudWatchException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [PutRule](#) in *AWS SDK for Java 2.x API Reference*.

## Send events

The following code example shows how to send Amazon CloudWatch Events events.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void putCWEVENTS(CloudWatchEventsClient cwe, String resourceArn ) {  
    try {  
  
        final String EVENT_DETAILS =  
            "{ \"key1\": \"value1\", \"key2\": \"value2\" }";  
  
        PutEventsRequestEntry requestEntry = PutEventsRequestEntry.builder()  
            .detail(EVENT_DETAILS)  
            .detailType("sampleSubmitted")  
            .resources(resourceArn)
```

```
.source("aws-sdk-java-cloudwatch-example")
.build();

PutEventsRequest request = PutEventsRequest.builder()
.entries(requestEntry)
.build();

cwe.putEvents(request);
System.out.println("Successfully put CloudWatch event");

} catch (CloudWatchException e) {
System.err.println(e.awsErrorDetails().errorMessage());
System.exit(1);
}
}
```

- For API details, see [PutEvents](#) in *AWS SDK for Java 2.x API Reference*.

## CloudWatch Logs examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon CloudWatch Logs.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3191\)](#)

## Actions

### Create a subscription filter

The following code example shows how to create an Amazon CloudWatch Logs subscription filter.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void putSubFilters(CloudWatchLogsClient cwl,
String filter,
String pattern,
String logGroup,
String functionArn) {

try {
PutSubscriptionFilterRequest request =
PutSubscriptionFilterRequest.builder()
.filterName(filter)
.filterPattern(pattern)
.logGroupName(logGroup)
.destinationArn(functionArn)
.build();

cwl.putSubscriptionFilter(request);
}
```

```
        System.out.printf(
            "Successfully created CloudWatch logs subscription filter %s",
            filter);

    } catch (CloudWatchLogsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutSubscriptionFilter](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a subscription filter

The following code example shows how to delete an Amazon CloudWatch Logs subscription filter.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteSubFilter(CloudWatchLogsClient logs, String filter, String
logGroup) {

    try {
        DeleteSubscriptionFilterRequest request =
DeleteSubscriptionFilterRequest.builder()
            .filterName(filter)
            .logGroupName(logGroup)
            .build();

        logs.deleteSubscriptionFilter(request);
        System.out.printf("Successfully deleted CloudWatch logs subscription filter
%s", filter);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteSubscriptionFilter](#) in *AWS SDK for Java 2.x API Reference*.

## Describe existing subscription filters

The following code example shows how to describe Amazon CloudWatch Logs existing subscription filters.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeFilters(CloudWatchLogsClient logs, String logGroup) {
```

```
try {
    boolean done = false;
    String newToken = null;

    while(!done) {
        DescribeSubscriptionFiltersResponse response;
        if (newToken == null) {
            DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                .logGroupName(logGroup)
                .limit(1).build();

            response = logs.describeSubscriptionFilters(request);
        } else {
            DescribeSubscriptionFiltersRequest request =
DescribeSubscriptionFiltersRequest.builder()
                .nextToken(newToken)
                .logGroupName(logGroup)
                .limit(1).build();
            response = logs.describeSubscriptionFilters(request);
        }

        for(SubscriptionFilter filter : response.subscriptionFilters()) {
            System.out.printf("Retrieved filter with name %s, " + "pattern %s "
+ "and destination arn %s",
                filter.filterName(),
                filter.filterPattern(),
                filter.destinationArn());
        }

        if(response.nextToken() == null) {
            done = true;
        } else {
            newToken = response.nextToken();
        }
    }

} catch (CloudWatchException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
System.out.printf("Done");
}
```

- For API details, see [DescribeSubscriptionFilters](#) in *AWS SDK for Java 2.x API Reference*.

## Amazon Cognito Identity Provider examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Cognito Identity Provider.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3194\)](#)
- [Scenarios \(p. 3199\)](#)

## Actions

### Confirm a user

The following code example shows how to confirm an Amazon Cognito user.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code, String userName) {

    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ConfirmSignUp in AWS SDK for Java 2.x API Reference](#).

### Get a token to associate an MFA application with a user

The following code example shows how to get a token to associate an MFA application with an Amazon Cognito user.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {

    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
    .session(session)
    .build();

    AssociateSoftwareTokenResponse tokenResponse =
identityProviderClient.associateSoftwareToken(softwareTokenRequest) ;
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}
```

- For API details, see [AssociateSoftwareToken](#) in *AWS SDK for Java 2.x API Reference*.

## Get information about a user

The following code example shows how to get information about an Amazon Cognito user.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName, String poolId) {

    try {
        Admin GetUserRequest userRequest = Admin GetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        Admin GetUserResponse response =
identityProviderClient.admin GetUser(userRequest);
        System.out.println("User status "+response.userStatusAsString());

    } catch (CognitoIdentityProviderException e){
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [Admin GetUser](#) in *AWS SDK for Java 2.x API Reference*.

## List users

The following code example shows how to list Amazon Cognito users.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllUsers(CognitoIdentityProviderClient cognitoClient, String
userPoolId ) {
    try {
        ListUsersRequest usersRequest = ListUsersRequest.builder()
            .userPoolId(userPoolId)
            .build();

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created " + user.userCreateDate() );
        });

    } catch (CognitoIdentityProviderException e){
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    // Shows how to list users by using a filter.
    public static void listUsersFilter(CognitoIdentityProviderClient cognitoClient,
String userPoolId ) {

    try {
        String filter = "email = \"tblue@noserver.com\"";
        ListUsersRequest usersRequest = ListUsersRequest.builder()
            .userPoolId(userPoolId)
            .filter(filter)
            .build();

        ListUsersResponse response = cognitoClient.listUsers(usersRequest);
        response.users().forEach(user -> {
            System.out.println("User with filter applied " + user.username() + " "
Status " + user.userStatus() + " Created " + user.userCreateDate() );
        });
    } catch (CognitoIdentityProviderException e){
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Java 2.x API Reference*.

## Resend a confirmation code

The following code example shows how to resend an Amazon Cognito confirmation code.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName) {

    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
            .clientId(clientId)
            .username(userName)
            .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is
"+response.codeDeliveryDetails().deliveryMediumAsString());

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ResendConfirmationCode](#) in *AWS SDK for Java 2.x API Reference*.

## Respond to an authentication challenge

The following code example shows how to respond to an Amazon Cognito authentication challenge.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Respond to an authentication challenge.
public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient, String userName, String clientId, String mfaCode, String
session) {

    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    RespondToAuthChallengeRequest respondToAuthChallengeRequest =
RespondToAuthChallengeRequest.builder()
        .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
        .clientId(clientId)
        .challengeResponses(challengeResponses)
        .session(session)
        .build();

    RespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient.respondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()" +
respondToAuthChallengeResult.authenticationResult());
}
```

- For API details, see [AdminRespondToAuthChallenge](#) in *AWS SDK for Java 2.x API Reference*.

## Sign up a user

The following code example shows how to sign up a user with Amazon Cognito.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName, String password, String email) {

    AttributeType userAttrs = AttributeType.builder()
        .name("email")
        .value(email)
        .build();

    List<AttributeType> userAttrsList = new ArrayList<>();
```

```
        userAttrsList.add(userAttrs);

        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                .userAttributes(userAttrsList)
                .username(userName)
                .clientId(clientId)
                .password(password)
                .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

        } catch(CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [SignUp](#) in *AWS SDK for Java 2.x API Reference*.

## Start authentication with administrator credentials

The following code example shows how to start authentication with Amazon Cognito and administrator credentials.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static InitiateAuthResponse initiateAuth(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName, String password) {

    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        InitiateAuthRequest authRequest = InitiateAuthRequest.builder()
            .clientId(clientId)
            .authParameters(authParameters)
            .authFlow(AuthFlowType.USER_PASSWORD_AUTH)
            .build();

        InitiateAuthResponse response =
identityProviderClient.initiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName() );
        return response;

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- For API details, see [AdminInitiateAuth](#) in *AWS SDK for Java 2.x API Reference*.

## Verify an MFA application with a user

The following code example shows how to verify an MFA application with an Amazon Cognito user.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient identityProviderClient,
String session, String code) {

    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
            .userCode(code)
            .session(session)
            .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is "
+verifyResponse.statusAsString());

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [VerifySoftwareToken](#) in *AWS SDK for Java 2.x API Reference*.

## Scenarios

### Sign up a user with a user pool that requires MFA

The following code example shows how to:

- Sign up a user with a user name, password, and email address.
- Confirm the user from a code sent in email.
- Set up multi-factor authentication by associating an MFA application with the user.
- Sign in by using a password and an MFA code.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
 * Before running this Java V2 code example, set up your development environment,
including your credentials.
*
* For more information, see the following documentation:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

```

```

/*
 * TIP: To set up the required user pool, run the AWS Cloud Development
 * Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
 * cognito_scenario_user_pool_with_mfa.
 */
/* This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the initiateAuth to sign in. This results in being prompted to set up
 * TOTP (time-based one-time password). (The response is "ChallengeName": "MFA_SETUP").
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.
 * This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being prompted to
 * submit a TOTP (Response: "ChallengeName": "SOFTWARE_TOKEN_MFA").
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws NoSuchAlgorithmException,
    InvalidKeyException {

        final String usage = "\n" +
            "Usage:\n" +
            "      <clientId> <poolId>\n\n" +
            "Where:\n" +
            "      clientId - The app client Id value that you can get from the AWS CDK
script.\n\n" +
            "      poolId - The pool Id that you can get from the AWS CDK script. \n\n" ;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("*** Enter your password");
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();

        System.out.println("1. Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);
        System.out.println(DASHES);
    }
}

```

```
System.out.println(DASHES);
System.out.println("2. Getting " + userName + " in the user pool");
getAdminUser(identityProviderClient, userName, poolId);

System.out.println("*** Conformation code sent to " + userName + ". Would you
like to send a new code? (Yes/No)");
System.out.println(DASHES);

System.out.println(DASHES);
String ans = in.nextLine();

if (ans.compareTo("Yes") == 0) {
    resendConfirmationCode(identityProviderClient, clientId, userName);
    System.out.println("3. Sending a new confirmation code");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Enter confirmation code that was emailed");
String code = in.nextLine();
confirmSignUp(identityProviderClient, clientId, code, userName);
System.out.println("Rechecking the status of " + userName + " in the user
pool");
getAdminUser(identityProviderClient, userName, poolId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Invokes the initiateAuth to sign in");
InitiateAuthResponse authResponse = initiateAuth(identityProviderClient,
clientId, userName, password) ;
String mySession = authResponse.session() ;
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Invokes the AssociateSoftwareToken method to generate a
TOTP key");
String newSession = getSecretForAppMFA(identityProviderClient, mySession);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
String myCode = in.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Verify the TOTP and register for MFA");
verifyTOTP(identityProviderClient, newSession, myCode);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
String mfaCode = in.nextLine();
InitiateAuthResponse authResponse1 = initiateAuth(identityProviderClient,
clientId, userName, password);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Invokes the AdminRespondToAuthChallenge");
String session2 = authResponse1.session();
adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient, String userName, String clientId, String mfaCode, String
session) {

    System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
    Map<String, String> challengeResponses = new HashMap<>();

    challengeResponses.put("USERNAME", userName);
    challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

    RespondToAuthChallengeRequest respondToAuthChallengeRequest =
RespondToAuthChallengeRequest.builder()
        .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
        .clientId(clientId)
        .challengeResponses(challengeResponses)
        .session(session)
        .build();

    RespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient.respondToAuthChallenge(respondToAuthChallengeRequest);
    System.out.println("respondToAuthChallengeResult.getAuthenticationResult()" +
respondToAuthChallengeResult.authenticationResult());
}

// Verify the TOTP and register for MFA.
public static void verifyTOTP(CognitoIdentityProviderClient identityProviderClient,
String session, String code) {

    try {
        VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
        .userCode(code)
        .session(session)
        .build();

        VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
        System.out.println("The status of the token is "
+verifyResponse.statusAsString());

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static InitiateAuthResponse initiateAuth(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName, String password) {

    try {
        Map<String, String> authParameters = new HashMap<>();
        authParameters.put("USERNAME", userName);
        authParameters.put("PASSWORD", password);

        InitiateAuthRequest authRequest = InitiateAuthRequest.builder()
            .clientId(clientId)
            .authParameters(authParameters)
            .authFlow(AuthFlowType.USER_PASSWORD_AUTH)
            .build();
    }
}
```

```
        InitiateAuthResponse response =
identityProviderClient.initiateAuth(authRequest);
        System.out.println("Result Challenge is : " + response.challengeName() );
        return response;

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}

public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {

    AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
    .session(session)
    .build();

    AssociateSoftwareTokenResponse tokenResponse =
identityProviderClient.associateSoftwareToken(softwareTokenRequest) ;
    String secretCode = tokenResponse.secretCode();
    System.out.println("Enter this token into Google Authenticator");
    System.out.println(secretCode);
    return tokenResponse.session();
}

public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code, String userName) {

    try {
        ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
            .clientId(clientId)
            .confirmationCode(code)
            .username(userName)
            .build();

        identityProviderClient.confirmSignUp(signUpRequest);
        System.out.println(userName + " was confirmed");

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName) {

    try {
        ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
    .clientId(clientId)
    .username(userName)
    .build();

        ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
        System.out.println("Method of delivery is
"+response.codeDeliveryDetails().deliveryMediumAsString());

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        }

    public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName, String password, String email) {

    AttributeType userAttrs = AttributeType.builder()
        .name("email")
        .value(email)
        .build();

    List<AttributeType> userAttrsList = new ArrayList<>();
    userAttrsList.add(userAttrs);

    try {
        SignUpRequest signUpRequest = SignUpRequest.builder()
            .userAttributes(userAttrsList)
            .username(userName)
            .clientId(clientId)
            .password(password)
            .build();

        identityProviderClient.signUp(signUpRequest);
        System.out.println("User has been signed up ");

    } catch(CognitoIdentityProviderException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName, String poolId) {

    try {
        Admin GetUserRequest userRequest = Admin GetUserRequest.builder()
            .username(userName)
            .userPoolId(poolId)
            .build();

        Admin GetUserResponse response =
identityProviderClient.admin GetUser(userRequest);
        System.out.println("User status "+response.userStatusAsString());

    } catch (CognitoIdentityProviderException e){
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.

- [Admin GetUser](#)
- [Admin Initiate Auth](#)
- [Admin Respond To Auth Challenge](#)
- [Associate Software Token](#)
- [Confirm Device](#)
- [Confirm Sign Up](#)
- [Initiate Auth](#)
- [List Users](#)

- [ResendConfirmationCode](#)
- [RespondToAuthChallenge](#)
- [SignUp](#)
- [VerifySoftwareToken](#)

## Amazon Comprehend examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Comprehend.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3205\)](#)

## Actions

### Create a document classifier

The following code example shows how to create an Amazon Comprehend document classifier.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createDocumentClassifier(ComprehendClient comClient, String
dataAccessRoleArn, String s3Uri, String documentClassifierName){

    try {
        DocumentClassifierInputDataConfig config =
DocumentClassifierInputDataConfig.builder()
            .s3Uri(s3Uri)
            .build();

        CreateDocumentClassifierRequest createDocumentClassifierRequest =
CreateDocumentClassifierRequest.builder()
            .documentClassifierName(documentClassifierName)
            .dataAccessRoleArn(dataAccessRoleArn)
            .languageCode("en")
            .inputDataConfig(config)
            .build();

        CreateDocumentClassifierResponse createDocumentClassifierResult =
comClient.createDocumentClassifier(createDocumentClassifierRequest);
        String documentClassifierArn =
createDocumentClassifierResult.documentClassifierArn();
        System.out.println("Document Classifier ARN: " + documentClassifierArn);

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateDocumentClassifier](#) in *AWS SDK for Java 2.x API Reference*.

## Detect entities in a document

The following code example shows how to detect entities in a document with Amazon Comprehend.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectAllEntities(ComprehendClient comClient, String text) {  
  
    try {  
        DetectEntitiesRequest detectEntitiesRequest =  
DetectEntitiesRequest.builder()  
            .text(text)  
            .languageCode("en")  
            .build();  
  
        DetectEntitiesResponse detectEntitiesResult =  
comClient.detectEntities(detectEntitiesRequest);  
        List<Entity> entList = detectEntitiesResult.entities();  
        for (Entity entity : entList) {  
            System.out.println("Entity text is " + entity.text());  
        }  
  
    } catch (ComprehendException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DetectEntities](#) in *AWS SDK for Java 2.x API Reference*.

## Detect key phrases in a document

The following code example shows how to detect key phrases in a document with Amazon Comprehend.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectAllKeyPhrases(ComprehendClient comClient, String text) {  
  
    try {  
        DetectKeyPhrasesRequest detectKeyPhrasesRequest =  
DetectKeyPhrasesRequest.builder()  
            .text(text)  
            .languageCode("en")  
            .build();  
  
        DetectKeyPhrasesResponse detectKeyPhrasesResult =  
comClient.detectKeyPhrases(detectKeyPhrasesRequest);  
    } catch (ComprehendException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

```
        List<KeyPhrase> phraseList = detectKeyPhrasesResult.keyPhrases();
        for (KeyPhrase keyPhrase : phraseList) {
            System.out.println("Key phrase text is " + keyPhrase.text());
        }

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DetectKeyPhrases](#) in *AWS SDK for Java 2.x API Reference*.

## Detect syntactical elements of a document

The following code example shows how to detect syntactical elements of a document with Amazon Comprehend.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectAllSyntax(ComprehendClient comClient, String text){

    try {
        DetectSyntaxRequest detectSyntaxRequest = DetectSyntaxRequest.builder()
            .text(text)
            .languageCode("en")
            .build();

        DetectSyntaxResponse detectSyntaxResult =
comClient.detectSyntax(detectSyntaxRequest);
        List<SyntaxToken> syntaxTokens = detectSyntaxResult.syntaxTokens();
        for (SyntaxToken token : syntaxTokens) {
            System.out.println("Language is " + token.text());
            System.out.println("Part of speech is " +
token.partOfSpeech().tagAsString());
        }

    } catch (ComprehendException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DetectSyntax](#) in *AWS SDK for Java 2.x API Reference*.

## Detect the dominant language in a document

The following code example shows how to detect the dominant language in a document with Amazon Comprehend.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectTheDominantLanguage(ComprehendClient comClient, String text){  
  
    try {  
        DetectDominantLanguageRequest request =  
DetectDominantLanguageRequest.builder()  
            .text(text)  
            .build();  
  
        DetectDominantLanguageResponse resp =  
comClient.detectDominantLanguage(request);  
        List<DominantLanguage> allLanList = resp.languages();  
        for (DominantLanguage lang : allLanList) {  
            System.out.println("Language is " + lang.languageCode());  
        }  
  
    } catch (ComprehendException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DetectDominantLanguage in AWS SDK for Java 2.x API Reference](#).

## Detect the sentiment of a document

The following code example shows how to detect the sentiment of a document with Amazon Comprehend.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectSentiments(ComprehendClient comClient, String text){  
  
    try {  
        DetectSentimentRequest detectSentimentRequest =  
DetectSentimentRequest.builder()  
            .text(text)  
            .languageCode("en")  
            .build();  
  
        DetectSentimentResponse detectSentimentResult =  
comClient.detectSentiment(detectSentimentRequest);  
        System.out.println("The Neutral value is "  
+detectSentimentResult.sentimentScore().neutral());  
  
    } catch (ComprehendException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DetectSentiment in AWS SDK for Java 2.x API Reference](#).

## DynamoDB examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon DynamoDB.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3209\)](#)
- [Scenarios \(p. 3219\)](#)

## Actions

### Create a table

The following code example shows how to create a DynamoDB table.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createTable(DynamoDbClient ddb, String tableName, String key) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    CreateTableRequest request = CreateTableRequest.builder()
        .attributeDefinitions(AttributeDefinition.builder()
            .attributeName(key)
            .attributeType(ScalarAttributeType.S)
            .build())
        .keySchema(KeySchemaElement.builder()
            .attributeName(key)
            .keyType(KeyType.HASH)
            .build())
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
        .tableName(tableName)
        .build();

    String newTable = "";
    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        // Wait until the Amazon DynamoDB table is created.
        WaiterResponse<DescribeTableResponse> waiterResponse =
        dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        newTable = response.tableDescription().tableName();
        return newTable;
    } catch (DynamoDbException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateTable](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a table

The following code example shows how to delete a DynamoDB table.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}
```

- For API details, see [DeleteTable](#) in *AWS SDK for Java 2.x API Reference*.

## Delete an item from a table

The following code example shows how to delete an item from a DynamoDB table.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal) {
    HashMap<String,AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    DeleteItemRequest deleteReq = DeleteItemRequest.builder()
        .tableName(tableName)
        .key(keyToGet)
        .build();
}
```

```
        try {
            ddb.deleteItem(deleteReq);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see [DeleteItem](#) in *AWS SDK for Java 2.x API Reference*.

## Get an item from a table

The following code example shows how to get an item from a DynamoDB table.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Gets an item from a table by using the `DynamoDbClient`.

```
public static int queryTable(DynamoDbClient ddb, String tableName, String
partitionKeyName, String partitionKeyVal, String partitionAlias) {

    // Set up an alias for the partition key name in case it's a reserved word.
    HashMap<String, String> attrNameAlias = new HashMap<String, String>();
    attrNameAlias.put(partitionAlias, partitionKeyName);

    // Set up mapping of the partition name with the value.
    HashMap<String, AttributeValue> attrValues = new HashMap<>();

    attrValues.put(":" + partitionKeyName, AttributeValue.builder()
        .s(partitionKeyVal)
        .build());

    QueryRequest queryReq = QueryRequest.builder()
        .tableName(tableName)
        .keyConditionExpression(partitionAlias + " = :" + partitionKeyName)
        .expressionAttributeNames(attrNameAlias)
        .expressionAttributeValues(attrValues)
        .build();

    try {
        QueryResponse response = ddb.query(queryReq);
        return response.count();
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return -1;
}
```

- For API details, see [GetItem](#) in *AWS SDK for Java 2.x API Reference*.

## Get information about a table

The following code example shows how to get information about a DynamoDB table.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDynamoDBTable(DynamoDbClient ddb, String tableName) {  
  
    DescribeTableRequest request = DescribeTableRequest.builder()  
        .tableName(tableName)  
        .build();  
  
    try {  
        TableDescription tableInfo = ddb.describeTable(request).table();  
        if (tableInfo != null) {  
            System.out.format("Table name : %s\n", tableInfo.tableName());  
            System.out.format("Table ARN : %s\n", tableInfo.tableArn());  
            System.out.format("Status : %s\n", tableInfo.tableStatus());  
            System.out.format("Item count : %d\n",  
                tableInfo.itemCount().longValue());  
            System.out.format("Size (bytes): %d\n",  
                tableInfo.tableSizeBytes().longValue());  
  
            ProvisionedThroughputDescription throughputInfo =  
                tableInfo.provisionedThroughput();  
            System.out.println("Throughput");  
            System.out.format(" Read Capacity : %d\n",  
                throughputInfo.readCapacityUnits().longValue());  
            System.out.format(" Write Capacity: %d\n",  
                throughputInfo.writeCapacityUnits().longValue());  
  
            List<AttributeDefinition> attributes =  
                tableInfo.attributeDefinitions();  
            System.out.println("Attributes");  
  
            for (AttributeDefinition a : attributes) {  
                System.out.format(" %s (%s)\n", a.attributeName(),  
                    a.attributeType());  
            }  
        }  
    } catch (DynamoDbException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
    System.out.println("\nDone!");  
}
```

- For API details, see [DescribeTable](#) in *AWS SDK for Java 2.x API Reference*.

## List tables

The following code example shows how to list DynamoDB tables.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllTables(DynamoDbClient ddb){
```

```
boolean moreTables = true;
String lastName = null;

while(moreTables) {
    try {
        ListTablesResponse response = null;
        if (lastName == null) {
            ListTablesRequest request = ListTablesRequest.builder().build();
            response = ddb.listTables(request);
        } else {
            ListTablesRequest request = ListTablesRequest.builder()
                .exclusiveStartTableName(lastName).build();
            response = ddb.listTables(request);
        }

        List<String> tableNames = response.tableNames();
        if (tableNames.size() > 0) {
            for (String curName : tableNames) {
                System.out.format("* %s\n", curName);
            }
        } else {
            System.out.println("No tables found!");
            System.exit(0);
        }

        lastName = response.lastEvaluatedTableName();
        if (lastName == null) {
            moreTables = false;
        }
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
System.out.println("\nDone!");
}
```

- For API details, see [ListTables](#) in *AWS SDK for Java 2.x API Reference*.

## Put an item in a table

The following code example shows how to put an item in a DynamoDB table.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Puts an item into a table by using the enhanced client.

```
public static void putRecord(DynamoDbEnhancedClient enhancedClient) {
    try {
        DynamoDbTable<Customer> custTable = enhancedClient.table("Customer",
TableSchema.fromBean(Customer.class));

        // Create an Instant value.
        LocalDate localDate = LocalDate.parse("2020-04-07");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);
```

```
// Populate the Table.  
Customer custRecord = new Customer();  
custRecord.setCustName("Tom red");  
custRecord.setId("id101");  
custRecord.setEmail("tred@noserver.com");  
custRecord.setRegistrationDate(instant) ;  
  
// Put the customer data into an Amazon DynamoDB table.  
custTable.putItem(custRecord);  
  
} catch (DynamoDbException e) {  
    System.err.println(e.getMessage());  
    System.exit(1);  
}  
System.out.println("Customer data added to the table with id id101");  
}
```

- For API details, see [PutItem](#) in *AWS SDK for Java 2.x API Reference*.

## Query a table

The following code example shows how to query a DynamoDB table.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Queries a table by using the enhanced client.

```
public static String queryTable(DynamoDbEnhancedClient enhancedClient) {  
  
    try{  
        DynamoDbTable<Customer> mappedTable = enhancedClient.table("Customer",  
TableSchema.fromBean(Customer.class));  
        QueryConditional queryConditional =  
QueryConditional.keyEqualTo(Key.builder()  
            .partitionValue("id101")  
            .build());  
  
        // Get items in the table and write out the ID value.  
        Iterator<Customer> results =  
mappedTable.query(queryConditional).items().iterator();  
        String result="";  
  
        while (results.hasNext()) {  
            Customer rec = results.next();  
            result = rec.getId();  
            System.out.println("The record id is "+result);  
        }  
        return result;  
  
    } catch (DynamoDbException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

Queries a table by using the enhanced client and a secondary index.

```
public static void queryIndex(DynamoDbClient ddb, String tableName) {  
  
    try {  
        // Create a DynamoDbEnhancedClient and use the DynamoDbClient object.  
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()  
            .dynamoDbClient(ddb)  
            .build();  
  
        //Create a DynamoDbTable object based on Movies.  
        DynamoDbTable<Movies> table = enhancedClient.table("Movies",  
TableSchema.fromBean(Movies.class));  
        String dateVal = "2013";  
  
        DynamoDbIndex<Movies> secIndex = enhancedClient.table("Movies",  
TableSchema.fromBean(Movies.class)).index("year-index");  
        AttributeValue attVal = AttributeValue.builder()  
            .n(dateVal)  
            .build();  
  
        // Create a QueryConditional object that's used in the query operation.  
        QueryConditional queryConditional = QueryConditional  
            .keyEqualTo(Key.builder().partitionValue(attVal)  
            .build());  
  
        // Get items in the table.  
        SdkIterable<Page<Movies>> results =  
secIndex.query(QueryEnhancedRequest.builder()  
            .queryConditional(queryConditional)  
            .limit(300)  
            .build());  
  
        // Display the results.  
        results.forEach(page -> {  
            List<Movies> allMovies = page.items();  
            for (Movies myMovies: allMovies) {  
                System.out.println("The movie title is " + myMovies.getTitle() + ".  
The year is " + myMovies.getYear());  
            }  
        });  
  
    } catch (DynamoDbException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

Queries a table by using the DynamoDbClient.

```
public static int queryTable(DynamoDbClient ddb, String tableName, String  
partitionKeyName, String partitionKeyVal, String partitionAlias) {  
  
    // Set up an alias for the partition key name in case it's a reserved word.  
    HashMap<String, String> attrNameAlias = new HashMap<String, String>();  
    attrNameAlias.put(partitionAlias, partitionKeyName);  
  
    // Set up mapping of the partition name with the value.  
    HashMap<String, AttributeValue> attrValues = new HashMap<>();  
  
    attrValues.put(": "+partitionKeyName, AttributeValue.builder()  
        .s(partitionKeyVal)  
        .build());
```

```
QueryRequest queryReq = QueryRequest.builder()
    .tableName(tableName)
    .keyConditionExpression(partitionAlias + " = :" + partitionKeyName)
    .expressionAttributeNames(attrNameAlias)
    .expressionAttributeValues(attrValues)
    .build();

try {
    QueryResponse response = ddb.query(queryReq);
    return response.count();

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return -1;
}
```

Queries a table by using the DynamoDbClient and a secondary index.

```
public static void queryIndex(DynamoDbClient ddb, String tableName) {

    try {
        Map<String, String> expressionAttributesNames = new HashMap<>();
        expressionAttributesNames.put("#year", "year");
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":yearValue",
        AttributeValue.builder().n("2013").build());

        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .indexName("year-index")
            .keyConditionExpression("#year = :yearValue")
            .expressionAttributeNames(expressionAttributesNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        System.out.println("== Movie Titles ==");
        QueryResponse response = ddb.query(request);
        response.items()
            .forEach(movie -> System.out.println(movie.get("title").s()));

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [Query in AWS SDK for Java 2.x API Reference](#).

## Scan a table

The following code example shows how to scan a DynamoDB table.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Scans an Amazon DynamoDB table by using the enhanced client.

```
public static void scan( DynamoDbEnhancedClient enhancedClient) {
    try{
        DynamoDbTable<Customer> custTable = enhancedClient.table("Customer",
TableSchema.fromBean(Customer.class));
        Iterator<Customer> results = custTable.scan().items().iterator();
        while (results.hasNext()) {
            Customer rec = results.next();
            System.out.println("The record id is "+rec.getId());
            System.out.println("The name is " +rec.getCustName());
        }
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

- For API details, see [Scan](#) in *AWS SDK for Java 2.x API Reference*.

## Update an item in a table

The following code example shows how to update an item in a DynamoDB table.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Updates an item located in a table by using the enhanced client.

```
public static String modifyItem(DynamoDbEnhancedClient enhancedClient, String
keyVal, String email) {

    try {

        DynamoDbTable<Customer> mappedTable = enhancedClient.table("Customer",
TableSchema.fromBean(Customer.class));
        Key key = Key.builder()
            .partitionValue(keyVal)
            .build();

        // Get the item by using the key and update the email value.
        Customer customerRec = mappedTable.getItem(r->r.key(key));
        customerRec.setEmail(email);
        mappedTable.updateItem(customerRec);
        return customerRec.getEmail();

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [UpdateItem](#) in *AWS SDK for Java 2.x API Reference*.

## Write a batch of items

The following code example shows how to write a batch of DynamoDB items.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Inserts many items into a table by using the enhanced client.

```
public static void putBatchRecords(DynamoDbEnhancedClient enhancedClient) {  
  
    try {  
        DynamoDbTable<Customer> customerMappedTable =  
enhancedClient.table("Customer", TableSchema.fromBean(Customer.class));  
        DynamoDbTable<Music> musicMappedTable = enhancedClient.table("Music",  
TableSchema.fromBean(Music.class));  
        LocalDate localDate = LocalDate.parse("2020-04-07");  
        LocalDateTime localDateTime = localDate.atStartOfDay();  
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);  
  
        Customer record2 = new Customer();  
        record2.setCustName("Fred Pink");  
        record2.setId("id110");  
        record2.setEmail("fredp@noserver.com");  
        record2.setRegistrationDate(instant) ;  
  
        Customer record3 = new Customer();  
        record3.setCustName("Susan Pink");  
        record3.setId("id120");  
        record3.setEmail("spink@noserver.com");  
        record3.setRegistrationDate(instant) ;  
  
        Customer record4 = new Customer();  
        record4.setCustName("Jerry orange");  
        record4.setId("id101");  
        record4.setEmail("jorange@noserver.com");  
        record4.setRegistrationDate(instant) ;  
  
        BatchWriteItemEnhancedRequest batchWriteItemEnhancedRequest =  
BatchWriteItemEnhancedRequest.builder()  
            .writeBatches(  
                WriteBatch.builder(Customer.class)           // add items to  
the Customer table  
                    .mappedTableResource(customerMappedTable)  
                    .addPutItem(builder -> builder.item(record2))  
                    .addPutItem(builder -> builder.item(record3))  
                    .addPutItem(builder -> builder.item(record4))  
                    .build(),  
                WriteBatch.builder(Music.class)           // delete an  
item from the Music table  
                    .mappedTableResource(musicMappedTable)  
                    .addDeleteItem(builder -> builder.key(  
                        Key.builder().partitionValue("Famous  
Band").build()))  
                    .build())  
            .build();  
  
        // Add three items to the Customer table and delete one item from the Music  
table  
        enhancedClient.batchWriteItem(batchWriteItemEnhancedRequest);  
    }  
}
```

```
        System.out.println("done");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [BatchWriteItem in AWS SDK for Java 2.x API Reference](#).

## Scenarios

### Get started using tables, items, and queries

The following code example shows how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.
- Delete a movie from the table.
- Delete the table.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a DynamoDB table.

```
// Create a table with a Sort key.
public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
    KeySchemaElement key = KeySchemaElement.builder()
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
```

```
.attributeName("title")
.keyType(KeyType.RANGE)
.build();

// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
.keySchema(tableKey)
.provisionedThroughput(ProvisionedThroughput.builder()
.readCapacityUnits(new Long(10))
.writeCapacityUnits(new Long(10))
.build())
.attributeDefinitions(attributeDefinitions)
.tableName(tableName)
.build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " +newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

```
// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String fileName)
throws IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

    DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0 ;
    while (iter.hasNext()) {
        // Only add 200 Movies to the table.
        if (t == 200)
            break ;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();
    }
}
```

```
Movies movies = new Movies();
movies.setYear(year);
movies.setTitle(title);
movies.setInfo(info);

// Put the data into the Amazon DynamoDB Movie table.
mappedTable.putItem(movies);
t++;
}
}
```

Get an item from a table.

```
public static void getItem(DynamoDbClient ddb) {

    HashMap<String,AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put("year", AttributeValue.builder()
        .n("1933")
        .build());

    keyToGet.put("title", AttributeValue.builder()
        .s("King Kong")
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName("Movies")
        .build();

    try {
        Map<String,AttributeValue> returnedItem = ddb.getItem(request).item();

        if (returnedItem != null) {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");

            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
                    returnedItem.get(key1).toString());
            }
        } else {
            System.out.format("No item found with the key %s!\n", "year");
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Full example.

```
/*
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:

```

```
/*
 * 1. Creates the Amazon DynamoDB Movie table with partition and sort key.
 * 2. Puts data into the Amazon DynamoDB table from a JSON document using the Enhanced
client.
 * 3. Gets data from the Movie table.
 * 4. Adds a new item.
 * 5. Updates an item.
 * 6. Uses a Scan to query items using the Enhanced client.
 * 7. Queries all items where the year is 2013 using the Enhanced Client.
 * 8. Deletes the table.
*/
```

```
public class Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws IOException {
        final String usage = "\n" +
            "Usage:\n" +
            "      <fileName>\n\n" +
            "Where:\n" +
            "      fileName - The path to the moviedata.json file that you can download
from the Amazon DynamoDB Developer Guide.\n" ;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = "Movies";
        String fileName = args[0];
        ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .credentialsProvider(credentialsProvider)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon DynamoDB example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Creating an Amazon DynamoDB table named Movies with a
key named year and a sort key named title.");
        createTable(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Loading data into the Amazon DynamoDB table.");
        loadData(ddb, tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Getting data from the Movie table.");
        getItem(ddb) ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Putting a record into the Amazon DynamoDB table.");
        putRecord(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Updating a record.");
        updateTableItem(ddb, tableName);
        System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("6. Scanning the Amazon DynamoDB table.");
scanMovies(ddb, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Querying the Movies released in 2013.");
queryTable(ddb);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Deleting the Amazon DynamoDB table.");
deleteDynamoDBTable(ddb, tableName);
System.out.println(DASHES);

ddb.close();
}

// Create a table with a Sort key.
public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
    KeySchemaElement key = KeySchemaElement.builder()
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE)
        .build();

    // Add KeySchemaElement objects to the list.
    tableKey.add(key);
    tableKey.add(key2);

    CreateTableRequest request = CreateTableRequest.builder()
        .keySchema(tableKey)
        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
        .attributeDefinitions(attributeDefinitions)
        .tableName(tableName)
        .build();

    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();
```

```
// Wait until the Amazon DynamoDB table is created.
WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " +newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Query the table.
public static void queryTable(DynamoDbClient ddb) {
try {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

    DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
    QueryConditional queryConditional = QueryConditional
        .keyEqualTo(Key.builder()
        .partitionValue(2013)
        .build());

    // Get items in the table and write out the ID value.
    Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
    String result="";

    while (results.hasNext()) {
        Movies rec = results.next();
        System.out.println("The title of the movie is "+rec.getTitle());
        System.out.println("The movie information is "+rec.getInfo());
    }

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Scan the table.
public static void scanMovies(DynamoDbClient ddb, String tableName) {
    System.out.println("***** Scanning all movies.\n");
    try{
        DynamoDbEnhancedClient enhancedClient =
DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        Iterator<Movies> results = custTable.scan().items().iterator();
        while (results.hasNext()) {
            Movies rec = results.next();
            System.out.println("The movie title is "+rec.getTitle());
            System.out.println("The movie year is " +rec.getYear());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String fileName)
throws IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

    DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0 ;
    while (iter.hasNext()) {
        // Only add 200 Movies to the table.
        if (t == 200)
            break ;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        Movies movies = new Movies();
        movies.setYear(year);
        movies.setTitle(title);
        movies.setInfo(info);

        // Put the data into the Amazon DynamoDB Movie table.
        mappedTable.putItem(movies);
        t++;
    }
}

// Update the record to include show only directors.
public static void updateTableItem(DynamoDbClient ddb, String tableName){
    HashMap<String,AttributeValue> itemKey = new HashMap<>();
    itemKey.put("year", AttributeValue.builder().n("1933").build());
    itemKey.put("title", AttributeValue.builder().s("King Kong").build());

    HashMap<String,AttributeValueUpdate> updatedValues = new HashMap<>();
    updatedValues.put("info", AttributeValueUpdate.builder()
        .value(AttributeValue.builder().s("{\"directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack\"]]").build())
        .action(AttributeAction.PUT)
        .build());

    UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(itemKey)
        .attributeUpdates(updatedValues)
        .build();

    try {
        ddb.updateItem(request);
    } catch (ResourceNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
        System.out.println("Item was updated!");
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    public static void putRecord(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));

            // Populate the Table.
            Movies record = new Movies();
            record.setYear(2020);
            record.setTitle("My Movie2");
            record.setInfo("no info");
            table.putItem(record);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Added a new movie to the table.");
    }

    public static void getItem(DynamoDbClient ddb) {

        HashMap<String,AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put("year", AttributeValue.builder()
            .n("1933")
            .build());

        keyToGet.put("title", AttributeValue.builder()
            .s("King Kong")
            .build());

        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName("Movies")
            .build();

        try {
            Map<String,AttributeValue> returnedItem = ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
```

```
        System.out.format("%s: %s\n", key1,
    returnedItem.get(key1).toString());
    }
} else {
    System.out.format("No item found with the key %s!\n", "year");
}

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## Query a table by using batches of PartiQL statements

The following code example shows how to:

- Get a batch of items by running multiple SELECT statements.
- Add a batch of items by running multiple INSERT statements.
- Update a batch of items by running multiple UPDATE statements.
- Delete a batch of items by running multiple DELETE statements.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public class ScenarioPartiQLBatch {

    public static void main(String [] args) throws IOException {

        String tableName = "MoviesPartiQLBatch";
        ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .credentialsProvider(credentialsProvider)
            .region(region)
            .build();
```

```
        System.out.println("***** Creating an Amazon DynamoDB table named "+tableName+
+" with a key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("***** Adding multiple records into the "+ tableName +
table using a batch command.");
        putRecordBatch(ddb);

        System.out.println("***** Updating multiple records using a batch command.");
        updateTableItemBatch(ddb);

        System.out.println("***** Deleting multiple records using a batch command.");
        deleteItemBatch(ddb);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .provisionedThroughput(ProvisionedThroughput.builder()
                .readCapacityUnits(new Long(10))
                .writeCapacityUnits(new Long(10))
                .build())
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
```

```
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " +newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void putRecordBatch(DynamoDbClient ddb) {
    String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?:",
'title' : ?, 'info' : ?}";
    try {
        // Create three movies to add to the Amazon DynamoDB table.
        // Set data for Movie 1.
        List<AttributeValue> parameters = new ArrayList<>();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        AttributeValue att3 = AttributeValue.builder()
            .s("No Information")
            .build();

        parameters.add(att1);
        parameters.add(att2);
        parameters.add(att3);

        BatchStatementRequest statementRequestMovie1 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parameters)
            .build();

        // Set data for Movie 2.
        List<AttributeValue> parametersMovie2 = new ArrayList<>();
        AttributeValue attMovie2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attMovie2A = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        AttributeValue attMovie2B = AttributeValue.builder()
            .s("No Information")
            .build();

        parametersMovie2.add(attMovie2);
        parametersMovie2.add(attMovie2A);
        parametersMovie2.add(attMovie2B);

        BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersMovie2)
            .build();
    }
}
```

```
// Set data for Movie 3.
List<AttributeValue> parametersMovie3 = new ArrayList<>();
AttributeValue attMovie3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attMovie3A = AttributeValue.builder()
    .s("My Movie 3")
    .build();

AttributeValue attMovie3B = AttributeValue.builder()
    .s("No Information")
    .build();

parametersMovie3.add(attMovie3);
parametersMovie3.add(attMovie3A);
parametersMovie3.add(attMovie3B);

BatchStatementRequest statementRequestMovie3 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersMovie3)
    .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
myBatchStatementList.add(statementRequestMovie1);
myBatchStatementList.add(statementRequestMovie2);
myBatchStatementList.add(statementRequestMovie3);

BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
    .statements(myBatchStatementList)
    .build();

BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
System.out.println("ExecuteStatement successful: " + response.toString());
System.out.println("Added new movies using a batch command.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void updateTableItemBatch(DynamoDbClient ddb){
    String sqlStatement = "UPDATE MoviesPartiQBatch SET info = 'directors\':"
    [\"Merian C. Cooper\", \"Ernest B. Schoedsack\" where year=? and title=?";
    List<AttributeValue> parametersRec1 = new ArrayList<>();

    // Update three records.
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2022"))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("My Movie 1")
        .build();

    parametersRec1.add(att1);
    parametersRec1.add(att2);

    BatchStatementRequest statementRequestRec1 = BatchStatementRequest.builder()
        .statement(sqlStatement)
        .parameters(parametersRec1)
```

```
.build();

// Update record 2.
List<AttributeValue> parametersRec2 = new ArrayList<>();
AttributeValue attRec2 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec2a = AttributeValue.builder()
    .s("My Movie 2")
    .build();

parametersRec2.add(attRec2);
parametersRec2.add(attRec2a);
BatchStatementRequest statementRequestRec2 = BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec2)
    .build();

// Update record 3.
List<AttributeValue> parametersRec3 = new ArrayList<>();
AttributeValue attRec3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec3a = AttributeValue.builder()
    .s("My Movie 3")
    .build();

parametersRec3.add(attRec3);
parametersRec3.add(attRec3a);
BatchStatementRequest statementRequestRec3 = BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec3)
    .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
myBatchStatementList.add(statementRequestRec1);
myBatchStatementList.add(statementRequestRec2);
myBatchStatementList.add(statementRequestRec3);

BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
    .statements(myBatchStatementList)
    .build();

try {
    BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
    System.out.println("ExecuteStatement successful: " + response.toString());
    System.out.println("Updated three movies using a batch command.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.out.println("Item was updated!");
}

public static void deleteItemBatch(DynamoDbClient ddb){
    String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and
title=?";
    List<AttributeValue> parametersRec1 = new ArrayList<>();

    // Specify three records to delete.
```

```
AttributeValue att1 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue att2 = AttributeValue.builder()
    .s("My Movie 1")
    .build();

parametersRec1.add(att1);
parametersRec1.add(att2);

BatchStatementRequest statementRequestRec1 = BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec1)
    .build();

// Specify record 2.
List<AttributeValue> parametersRec2 = new ArrayList<>();
AttributeValue attRec2 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec2a = AttributeValue.builder()
    .s("My Movie 2")
    .build();

parametersRec2.add(attRec2);
parametersRec2.add(attRec2a);
BatchStatementRequest statementRequestRec2 = BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec2)
    .build();

// Specify record 3.
List<AttributeValue> parametersRec3 = new ArrayList<>();
AttributeValue attRec3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec3a = AttributeValue.builder()
    .s("My Movie 3")
    .build();

parametersRec3.add(attRec3);
parametersRec3.add(attRec3a);

BatchStatementRequest statementRequestRec3 = BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec3)
    .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
myBatchStatementList.add(statementRequestRec1);
myBatchStatementList.add(statementRequestRec2);
myBatchStatementList.add(statementRequestRec3);

BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
    .statements(myBatchStatementList)
    .build();

try {
    ddb.batchExecuteStatement(batchRequest);
    System.out.println("Deleted three movies using a batch command.");
}
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName +" was successfully deleted!");
    }

    private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient ddb,
String statement, List<AttributeValue> parameters ) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
    }
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Java 2.x API Reference*.

## Query a table using PartiQL

The following code example shows how to:

- Get an item by running a SELECT statement.
- Add an item by running an INSERT statement.
- Update an item by running an UPDATE statement.
- Delete an item by running a DELETE statement.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public class ScenarioPartiQ {

    public static void main(String [] args) throws IOException {

        final String usage = "\n" +
            "Usage:\n" +
            "      <fileName>\n\n" +
            "Where:\n" +
            "      fileName - The path to the moviedata.json file that you can download
from the Amazon DynamoDB Developer Guide.\n" ;
```

```
if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String fileName = args[0];
String tableName = "MoviesPartiQ";
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .credentialsProvider(credentialsProvider)
    .region(region)
    .build();

System.out.println("***** Creating an Amazon DynamoDB table named MoviesPartiQ
with a key named year and a sort key named title.");
createTable(ddb, tableName);

System.out.println("***** Loading data into the MoviesPartiQ table.");
loadData(ddb, fileName);

System.out.println("***** Getting data from the MoviesPartiQ table.");
getItem(ddb);

System.out.println("***** Putting a record into the MoviesPartiQ table.");
putRecord(ddb);

System.out.println("***** Updating a record.");
updateTableItem(ddb);

System.out.println("***** Querying the movies released in 2013.");
queryTable(ddb);

System.out.println("***** Deleting the Amazon DynamoDB table.");
deleteDynamoDBTable(ddb, tableName);
ddb.close();
}

public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
    KeySchemaElement key = KeySchemaElement.builder()
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE) // Sort
        .build();
}
```

```
// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(new Long(10))
        .writeCapacityUnits(new Long(10))
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
    dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " +newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String fileName) throws IOException
{

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?, 'info' : ?}";
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0 ;
    List<AttributeValue> parameters = new ArrayList<>();
    while (iter.hasNext()) {

        // Add 200 movies to the table.
        if (t == 200)
            break ;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf(year))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s(title)
            .build();

        AttributeValue att3 = AttributeValue.builder()
```

```
        .s(info)
        .build();

    parameters.add(att1);
    parameters.add(att2);
    parameters.add(att3);

    // Insert the movie into the Amazon DynamoDB table.
    executeStatementRequest(ddb, sqlStatement, parameters);
    System.out.println("Added Movie " +title);

    parameters.remove(att1);
    parameters.remove(att2);
    parameters.remove(att3);
    t++;
}
}

public static void getItem(DynamoDbClient ddb) {

    String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n("2012")
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("The Perks of Being a Wallflower")
        .build();

    parameters.add(att1);
    parameters.add(att2);

    try {
        ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
        System.out.println("ExecuteStatement successful: "+ response.toString());

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void putRecord(DynamoDbClient ddb) {

    String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?, 'info' : ?}";
    try {
        List<AttributeValue> parameters = new ArrayList<>();

        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2020"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie")
            .build();

        AttributeValue att3 = AttributeValue.builder()
            .s("No Information")
            .build();

        parameters.add(att1);
        parameters.add(att2);
        parameters.add(att3);
    }
}
```

```
        executeStatementRequest(ddb, sqlStatement, parameters);
        System.out.println("Added new movie.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateTableItem(DynamoDbClient ddb){

    String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C. Cooper\", \"Ernest B. Schoedsack\"] WHERE year=? AND title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2013"))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("The East")
        .build();

    parameters.add(att1);
    parameters.add(att2);

    try {
        executeStatementRequest(ddb, sqlStatement, parameters);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Item was updated!");
}

// Query the table where the year is 2013.
public static void queryTable(DynamoDbClient ddb) {
    String sqlStatement = "SELECT * FROM MoviesPartiQ WHERE year = ? ORDER BY
year";
    try {

        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2013"))
            .build();
        parameters.add(att1);

        // Get items in the table and write out the ID value.
        ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
        System.out.println("ExecuteStatement successful: "+ response.toString());

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
```

```
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName +" was successfully deleted!");
}

private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient ddb,
String statement, List<AttributeValue> parameters ) {
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .statement(statement)
        .parameters(parameters)
        .build();

    return ddb.executeStatement(request);
}

private static void processResults(ExecuteStatementResponse executeStatementResult)
{
    System.out.println("ExecuteStatement successful: "+
executeStatementResult.toString());
}
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Java 2.x API Reference*.

## Amazon EC2 examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Elastic Compute Cloud.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Get started

#### Hello Amazon EC2

The following code examples show how to get started using Amazon Elastic Compute Cloud (Amazon EC2).

Java

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeEC2SecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
        .groupIds(groupId)
```

```
        .build();

        DescribeSecurityGroupsResponse response =
ec2.describeSecurityGroups(request);
        for(SecurityGroup group : response.securityGroups()) {
            System.out.printf(
                "Found Security Group with id %s, " +
                "vpc id %s " +
                "and description %s",
                group.groupId(),
                group.vpcId(),
                group.description());
        }

    } catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import boto3

def hello_ec2(ec2_resource):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Elastic Compute Cloud
    (Amazon EC2) resource and list the security groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ec2_resource: A Boto3 EC2 ServiceResource object. This object is a high-
level
                        resource that wraps the low-level EC2 service API.
    """
    print("Hello, Amazon EC2! Let's list up to 10 of your security groups:")
    for sg in ec2_resource.security_groups.limit(10):
        print(f"\t{sg.id}: {sg.group_name}")

if __name__ == '__main__':
    hello_ec2(boto3.resource('ec2'))
```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for Python (Boto3) API Reference*.

## Topics

- [Actions \(p. 3240\)](#)
- [Scenarios \(p. 3249\)](#)

## Actions

### Allocate an Elastic IP address

The following code example shows how to allocate an Elastic IP address for Amazon EC2.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String getAllocateAddress( Ec2Client ec2, String instanceId) {  
  
    try {  
        AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()  
            .domain(DomainType.VPC)  
            .build();  
  
        AllocateAddressResponse allocateResponse =  
ec2.allocateAddress(allocateRequest);  
        String allocationId = allocateResponse.allocationId();  
        AssociateAddressRequest associateRequest =  
AssociateAddressRequest.builder()  
            .instanceId(instanceId)  
            .allocationId(allocationId)  
            .build();  
  
        AssociateAddressResponse associateResponse =  
ec2.associateAddress(associateRequest);  
        return associateResponse.associationId();  
  
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [AllocateAddress in AWS SDK for Java 2.x API Reference](#).

### Associate an Elastic IP address with an instance

The following code example shows how to associate an Elastic IP address with an Amazon EC2 instance.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String associateAddress(Ec2Client ec2, String instanceId, String  
allocationId) {  
    try {  
        AssociateAddressRequest associateRequest =  
AssociateAddressRequest.builder()  
            .instanceId(instanceId)  
            .allocationId(allocationId)  
            .build();  
    }
```

```
        AssociateAddressResponse associateResponse =
    ec2.associateAddress(associateRequest);
    return associateResponse.associationId();

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [AssociateAddress](#) in *AWS SDK for Java 2.x API Reference*.

## Create a security group

The following code example shows how to create an Amazon EC2 security group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createEC2SecurityGroup( Ec2Client ec2, String groupName, String
groupDesc, String vpcId) {
try {

    CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
    .groupName(groupName)
    .description(groupDesc)
    .vpcId(vpcId)
    .build();

    CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);

    IpRange ipRange = IpRange.builder()
        .cidrIp("0.0.0.0/0").build();

    IpPermission ipPerm = IpPermission.builder()
        .ipProtocol("tcp")
        .toPort(80)
        .fromPort(80)
        .ipRanges(ipRange)
        .build();

    IpPermission ipPerm2 = IpPermission.builder()
        .ipProtocol("tcp")
        .toPort(22)
        .fromPort(22)
        .ipRanges(ipRange)
        .build();

    AuthorizeSecurityGroupIngressRequest authRequest =
        AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
            .ipPermissions(ipPerm, ipPerm2)
            .build();

    AuthorizeSecurityGroupIngressResponse authResponse =

```

```
        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.printf("Successfully added ingress policy to Security Group %s",
groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateSecurityGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Create a security key pair

The following code example shows how to create a security key pair for Amazon EC2.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createEC2KeyPair(Ec2Client ec2, String keyName ) {
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        ec2.createKeyPair(request);
        System.out.printf("Successfully created key pair named %s", keyName);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateKeyPair](#) in *AWS SDK for Java 2.x API Reference*.

## Create and run an instance

The following code example shows how to create and run an Amazon EC2 instance.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createEC2Instance(Ec2Client ec2, String name, String amiId ) {

    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
```

```
.maxCount(1)
.minCount(1)
.build();

RunInstancesResponse response = ec2.runInstances(runRequest);
String instanceId = response.instances().get(0).instanceId();
Tag tag = Tag.builder()
    .key("Name")
    .value(name)
    .build();

CreateTagsRequest tagRequest = CreateTagsRequest.builder()
    .resources(instanceId)
    .tags(tag)
    .build();

try {
    ec2.createTags(tagRequest);
    System.out.printf("Successfully started EC2 Instance %s based on AMI %s",
instanceId, amiId);
    return instanceId;
} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
```

- For API details, see [RunInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a security group

The following code example shows how to delete an Amazon EC2 security group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {

    try {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        ec2.deleteSecurityGroup(request);
        System.out.printf("Successfully deleted Security Group with id %s",
groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteSecurityGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a security key pair

The following code example shows how to delete an Amazon EC2 security key pair.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteKeys(Ec2Client ec2, String keyPair) {  
  
    try {  
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()  
            .keyName(keyPair)  
            .build();  
  
        ec2.deleteKeyPair(request);  
        System.out.printf("Successfully deleted key pair named %s", keyPair);  
  
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteKeyPair](#) in *AWS SDK for Java 2.x API Reference*.

## Describe instances

The following code example shows how to describe Amazon EC2 instances.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String describeEC2Instances( Ec2Client ec2, String newInstanceId) {  
    try {  
        String pubAddress = "";  
        boolean isRunning = false;  
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()  
            .instanceIds(newInstanceId)  
            .build();  
  
        while (!isRunning) {  
            DescribeInstancesResponse response = ec2.describeInstances(request);  
            String state =  
response.reservations().get(0).instances().get(0).state().name().name();  
            if (state.compareTo("RUNNING") == 0) {  
                System.out.println("Image id is " +  
response.reservations().get(0).instances().get(0).imageId());  
                System.out.println("Instance type is " +  
response.reservations().get(0).instances().get(0).instanceType());  
                System.out.println("Instance state is " +  
response.reservations().get(0).instances().get(0).state().name());  
                pubAddress =  
response.reservations().get(0).instances().get(0).publicIpAddress();  
                System.out.println("Instance address is " + pubAddress);  
            }  
        }  
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

```
        isRunning = true;
    }
    return pubAddress;
} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [DescribeInstances](#) in *AWS SDK for Java 2.x API Reference*.

### Disassociate an Elastic IP address from an instance

The following code example shows how to disassociate an Elastic IP address from an Amazon EC2 instance.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
        .associationId(associationId)
        .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DisassociateAddress](#) in *AWS SDK for Java 2.x API Reference*.

### Get data about a security group

The following code example shows how to get data about an Amazon EC2 security group.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
        .groupIds(groupId)
```

```
.build();

DescribeSecurityGroupsResponse response =
ec2.describeSecurityGroups(request);
for(SecurityGroup group : response.securityGroups()) {
    System.out.println( "Found Security Group with Id " +group.groupId() +
and group VPC "+ group.vpcId());
}

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for Java 2.x API Reference*.

## Get data about instance types

The following code example shows how to get data about Amazon EC2 instance types.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType="";
    try {
        List<Filter> filters = new ArrayList<>();
        Filter filter = Filter.builder()
            .name("processor-info.supported-architecture")
            .values("arm64")
            .build();

        filters.add(filter);
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .filters(filters)
            .maxResults(10)
            .build();

        DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
        List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
        for (InstanceTypeInfo type: instanceTypes) {
            System.out.println("The memory information of this type is
"+type.memoryInfo().sizeInMiB());
            System.out.println("Network information is
"+type.networkInfo().toString());
            instanceType = type.instanceType().toString();
        }

        return instanceType;
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
        return "";
    }
```

- For API details, see [DescribeInstanceTypes](#) in *AWS SDK for Java 2.x API Reference*.

## Release an Elastic IP address

The following code example shows how to release an Elastic IP address.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void releaseEC2Address(Ec2Client ec2, String allocId) {

    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        ec2.releaseAddress(request);
        System.out.printf("Successfully released elastic IP address %s", allocId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ReleaseAddress](#) in *AWS SDK for Java 2.x API Reference*.

## Start an instance

The following code example shows how to start an Amazon EC2 instance.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void startInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to run. This will
take a few minutes.");
    ec2.startInstances(request);
    DescribeInstancesRequest instanceRequest = DescribeInstancesRequest.builder()
```

```
.instanceIds(instanceId)
.build();

WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Successfully started instance "+instanceId);
}
```

- For API details, see [StartInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Stop an instance

The following code example shows how to stop an Amazon EC2 instance.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest = DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance "+instanceId);
}
```

- For API details, see [StopInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Terminate an instance

The following code example shows how to terminate an Amazon EC2 instance.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void terminateEC2( Ec2Client ec2, String instanceID) {
```

```
try{
    TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
        .instanceIds(instanceID)
        .build();

    TerminateInstancesResponse response = ec2.terminateInstances(ti);
    List<InstanceStateChange> list = response.terminatingInstances();
    for (InstanceStateChange sc : list) {
        System.out.println("The ID of the terminated instance is " +
sc.instanceId());
    }

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [TerminateInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Scenarios

### Get started with instances

The following code example shows how to:

- Create a key pair that is used to secure SSH communication between your computer and an EC2 instance.
- Create a security group that acts as a virtual firewall for your EC2 instances to control incoming and outgoing traffic.
- Find an Amazon Machine Image (AMI) and a compatible instance type.
- Create an instance that is created from the instance type and AMI you select, and is configured to use the security group and key pair created in this example.
- Stop and restart the instance.
- Create an Elastic IP address and associate it as a consistent IP address for your instance.
- Connect to your instance with SSH, using both its public IP address and your Elastic IP address.
- Clean up all of the resources created by this example.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
 * Before running this Java (v2) code example, set up your development environment,
including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This Java example performs the following tasks:
*
* 1. Creates an RSA key pair and saves the private key data as a .pem file.
* 2. Lists key pairs.
```

```
* 3. Creates a security group for the default VPC.  
* 4. Displays security group information.  
* 5. Gets a list of Amazon Linux 2 AMIs and selects one.  
* 6. Gets more information about the image.  
* 7. Gets a list of instance types that are compatible with the selected AMI's  
architecture.  
* 8. Creates an instance with the key pair, security group, AMI, and an instance type.  
* 9. Displays information about the instance.  
* 10. Stops the instance and waits for it to stop.  
* 11. Starts the instance and waits for it to start.  
* 12. Allocates an Elastic IP address and associates it with the instance.  
* 13. Displays SSH connection info for the instance.  
* 14. Disassociates and deletes the Elastic IP address.  
* 15. Terminates the instance and waits for it to terminate.  
* 16. Deletes the security group.  
* 17. Deletes the key pair.  
*/  
public class EC2Scenario {  
  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    public static void main(String[] args) throws InterruptedException {  
  
        final String usage = "\n" +  
            "Usage:\n" +  
            "  <keyName> <fileName> <groupName> <groupDesc> <vpcId>\n\n" +  
            "Where:\n" +  
            "  keyName - A key pair name (for example, TestKeyPair). \n\n" +  
            "  fileName - A file name where the key information is written to. \n\n" +  
+           "  groupName - The name of the security group. \n\n" +  
+           "  groupDesc - The description of the security group. \n\n" +  
+           "  vpcId - A VPC ID. You can get this value from the AWS Management  
Console. \n\n" ;  
  
        if (args.length != 5) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String keyName = args[0];  
        String fileName = args[1];  
        String groupName = args[2];  
        String groupDesc = args[3];  
        String vpcId = args[4];  
  
        Region region = Region.US_WEST_2;  
        Ec2Client ec2 = Ec2Client.builder()  
            .region(region)  
            .credentialsProvider(ProfileCredentialsProvider.create())  
            .build();  
  
        SsmClient ssmClient = SsmClient.builder()  
            .region(region)  
            .credentialsProvider(ProfileCredentialsProvider.create())  
            .build();  
  
        System.out.println(DASHES);  
        System.out.println("Welcome to the Amazon EC2 example scenario.");  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("1. Create an RSA key pair and save the private key material  
as a .pem file.");  
        createKeyPair(ec2, keyName, fileName);  
        System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("2. List key pairs.");
describeKeys(ec2);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Create a security group.");
String groupId = createSecurityGroup(ec2, groupName, groupDesc, vpcId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Display security group info for the newly created
security group.");
describeSecurityGroups(ec2, groupId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a list of Amazon Linux 2 AMIs and selects one with
amzn2 in the name.");
String instanceId = getParaValues(ssmClient);
System.out.println("The instance Id is "+instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Get more information about an amzn2 image.");
String amiValue = describeImage(ec2, instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of instance types.");
String instanceType = getInstanceTypes(ec2);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Create an instance.");
String newInstanceId = runInstance(ec2, instanceType, keyName, groupName,
amiValue );
System.out.println("The instance Id is "+newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Display information about the running instance. ");
String ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i "+fileName +"ec2-user@"+ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Stop the instance and use a waiter.");
stopInstance(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Start the instance and use a waiter.");
startInstance(ec2, newInstanceId);
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i "+fileName +"ec2-user@"+ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Allocate an Elastic IP address and associate it with
the instance.");
String allocationId = allocateAddress(ec2);
System.out.println("The allocation Id value is "+allocationId);
String associationId = associateAddress(ec2, newInstanceId, allocationId);
```

```
System.out.println("The associate Id value is "+associationId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Describe the instance again.");
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i "+fileName +"ec2-user@"+ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Disassociate and release the Elastic IP address.");
disassociateAddress(ec2, associationId);
releaseEC2Address(ec2, allocationId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Terminate the instance and use a waiter.");
terminateEC2(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Delete the security group.");
deleteEC2SecGroup(ec2, groupId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("17. Delete the key.");
deleteKeys(ec2, keyName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("You successfully completed the Amazon EC2 scenario.! ");
System.out.println(DASHES);
ec2.close();
}

public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id "
+groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateEC2(Ec2Client ec2, String instanceId) {
    try{
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();

        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to terminate.
This will take a few minutes.");
    }
}
```

```
        ec2.terminateInstances(ti);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceTerminated(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance "+instanceId);
        System.out.println(instanceId + " is terminated!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named "+keyPair);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void releaseEC2Address(Ec2Client ec2, String allocId) {
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address "+allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
        .associationId(associationId)
        .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
```

```
AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
    .instanceId(instanceId)
    .allocationId(allocationId)
    .build();

AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
return associateResponse.associationId();

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static String allocateAddress(Ec2Client ec2) {
try {
    AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
        .domain(DomainType.VPC)
        .build();

    AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
    return allocateResponse.allocationId();

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static void startInstance(Ec2Client ec2, String instanceId) {
Ec2Waiter ec2Waiter = Ec2Waiter.builder()
    .overrideConfiguration(b -> b.maxAttempts(100))
    .client(ec2)
    .build();

StartInstancesRequest request = StartInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

System.out.println("Use an Ec2Waiter to wait for the instance to run. This will
take a few minutes.");
ec2.startInstances(request);
DescribeInstancesRequest instanceRequest = DescribeInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();

WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println("Successfully started instance "+instanceId);
}

public static void stopInstance(Ec2Client ec2, String instanceId) {
Ec2Waiter ec2Waiter = Ec2Waiter.builder()
    .overrideConfiguration(b -> b.maxAttempts(100))
    .client(ec2)
    .build();
StopInstancesRequest request = StopInstancesRequest.builder()
    .instanceIds(instanceId)
    .build();
```

```
        System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
        ec2.stopInstances(request);
        DescribeInstancesRequest instanceRequest = DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully stopped instance "+instanceId);
    }

    public static String describeEC2Instances( Ec2Client ec2, String newInstanceId) {
        try {
            String pubAddress = "";
            boolean isRunning = false;
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .instanceIds(newInstanceId)
                .build();

            while (!isRunning) {
                DescribeInstancesResponse response = ec2.describeInstances(request);
                String state =
response.reservations().get(0).instances().get(0).state().name().name();
                if (state.compareTo("RUNNING") ==0) {
                    System.out.println("Image id is " +
response.reservations().get(0).instances().get(0).imageId());
                    System.out.println("Instance type is " +
response.reservations().get(0).instances().get(0).instanceType());
                    System.out.println("Instance state is " +
response.reservations().get(0).instances().get(0).state().name());
                    pubAddress =
response.reservations().get(0).instances().get(0).publicIpAddress();
                    System.out.println("Instance address is " + pubAddress);
                    isRunning = true;
                }
            }
            return pubAddress;
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String runInstance(Ec2Client ec2, String instanceType, String
keyName, String groupName, String amiId ) {
        try {
            RunInstancesRequest runRequest = RunInstancesRequest.builder()
                .instanceType(instanceType)
                .keyName(keyName)
                .securityGroups(groupName)
                .maxCount(1)
                .minCount(1)
                .imageId(amiId)
                .build();

            RunInstancesResponse response = ec2.runInstances(runRequest);
            String instanceId = response.instances().get(0).instanceId();
            System.out.println("Successfully started EC2 instance "+instanceId +" based
on AMI "+amiId);
            return instanceId;

        } catch (SsmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

```
        System.exit(1);
    }
    return "";
}

// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType="";
    try {
        List<Filter> filters = new ArrayList<>();
        Filter filter = Filter.builder()
            .name("processor-info.supported-architecture")
            .values("arm64")
            .build();

        filters.add(filter);
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .filters(filters)
            .maxResults(10)
            .build();

        DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
        List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
        for (InstanceTypeInfo type: instanceTypes) {
            System.out.println("The memory information of this type is
"+type.memoryInfo().sizeInMiB());
            System.out.println("Network information is
"+type.networkInfo().toString());
            instanceType = type.instanceType().toString();
        }

        return instanceType;
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Display the Description field that corresponds to the instance Id value.
public static String describeImage(Ec2Client ec2, String instanceId) {
    try {
        DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
            .imageIds(instanceId)
            .build();

        DescribeImagesResponse response = ec2.describeImages(imagesRequest);
        System.out.println("The description of the first image is
"+response.images().get(0).description());
        System.out.println("The name of the first image is
"+response.images().get(0).name());

        // Return the image Id value.
        return response.images().get(0).imageId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Get the Id value of an instance with amzn2 in the name.
```

```
public static String getParaValues(SsmClient ssmClient) {
    try {
        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
            .build();

        GetParametersByPathIterable responses =
ssmClient.getParametersByPathPaginator(parameterRequest);
        for (software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse
response : responses) {
            System.out.println("Test "+response.nextToken());
            List<Parameter> parameterList = response.parameters();
            for (Parameter para: parameterList) {
                System.out.println("The name of the para is: "+para.name());
                System.out.println("The type of the para is: "+para.type());
                if (filterName(para.name())) {
                    return para.value();
                }
            }
        }
    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Return true if the name has amzn2 in it. For example:
// /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-gp2
private static boolean filterName(String name) {
    String[] parts = name.split("/");
    String myValue = parts[4];
    return myValue.contains("amzn2");
}

public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        DescribeSecurityGroupsResponse response =
ec2.describeSecurityGroups(request);
        for(SecurityGroup group : response.securityGroups()) {
            System.out.println( "Found Security Group with Id " +group.groupId() +
and group VPC "+ group.vpcId());
        }
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String createSecurityGroup( Ec2Client ec2, String groupName, String
groupDesc, String vpcId) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();
    }
```

```
CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);
IpRange ipRange = IpRange.builder()
    .cidrIp("0.0.0.0/0").build();

IpPermission ipPerm = IpPermission.builder()
    .ipProtocol("tcp")
    .toPort(80)
    .fromPort(80)
    .ipRanges(ipRange)
    .build();

IpPermission ipPerm2 = IpPermission.builder()
    .ipProtocol("tcp")
    .toPort(22)
    .fromPort(22)
    .ipRanges(ipRange)
    .build();

AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
    .groupName(groupName)
    .ipPermissions(ipPerm, ipPerm2)
    .build();

ec2.authorizeSecurityGroupIngress(authRequest);
System.out.println("Successfully added ingress policy to security group
"+groupName);
return resp.groupId();

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static void describeKeys( Ec2Client ec2){
try {
    DescribeKeyPairsResponse response = ec2.describeKeyPairs();
    response.keyPairs().forEach(keyPair -> System.out.printf(
        "Found key pair with name %s " +
        "and fingerprint %s",
        keyPair.keyName(),
        keyPair.keyFingerprint())
    );
}

} catch (Ec2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void createKeyPair(Ec2Client ec2, String keyName, String fileName) {
try {
    CreateKeyPairRequest request = CreateKeyPairRequest.builder()
        .keyName(keyName)
        .build();

    CreateKeyPairResponse response = ec2.createKeyPair(request);
    String content = response.keyMaterial();
    BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
    writer.write(content);
    writer.close();
    System.out.println("Successfully created key pair named "+keyName);
}
```

```
        } catch (Ec2Exception | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [AllocateAddress](#)
  - [AssociateAddress](#)
  - [AuthorizeSecurityGroupIngress](#)
  - [CreateKeyPair](#)
  - [CreateSecurityGroup](#)
  - [DeleteKeyPair](#)
  - [DeleteSecurityGroup](#)
  - [DescribeImages](#)
  - [DescribeInstanceTypes](#)
  - [DescribeInstances](#)
  - [DescribeKeyPairs](#)
  - [DescribeSecurityGroups](#)
  - [DisassociateAddress](#)
  - [ReleaseAddress](#)
  - [RunInstances](#)
  - [StartInstances](#)
  - [StopInstances](#)
  - [TerminateInstances](#)

## Amazon EC2 Auto Scaling examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon EC2 Auto Scaling.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3259\)](#)
- [Scenarios \(p. 3266\)](#)

## Actions

### Create a group

The following code example shows how to create an Auto Scaling group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
                                         String groupName,
                                         String launchTemplateName,
                                         String serviceLinkedRoleARN,
                                         String vpcZoneId) {

    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .availabilityZones("us-east-1a")
            .launchTemplate(templateSpecification)
            .maxSize(1)
            .minSize(1)
            .vpcZoneIdentifier(vpcZoneId)
            .serviceLinkedRoleARN(serviceLinkedRoleARN)
            .build();

        autoScalingClient.createAutoScalingGroup(request);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter.waitUntilGroupExists(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateAutoScalingGroup](#) in [AWS SDK for Java 2.x API Reference](#).

## Delete a group

The following code example shows how to delete an Auto Scaling group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
                                         String groupName) {
    try {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
```

```
        .build() ;  
  
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest) ;  
        System.out.println("You successfully deleted "+groupName);  
  
    } catch (AutoScalingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteAutoScalingGroup](#) in *AWS SDK for Java 2.x API Reference*.

### Disable metrics collection for a group

The following code example shows how to disable CloudWatch metrics collection for an Auto Scaling group.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void disableMetricsCollection(AutoScalingClient autoScalingClient,  
String groupName) {  
    try {  
        DisableMetricsCollectionRequest disableMetricsCollectionRequest =  
DisableMetricsCollectionRequest.builder()  
            .autoScalingGroupName(groupName)  
            .metrics("GroupMaxSize")  
            .build();  
  
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);  
        System.out.println("The disable metrics collection operation was  
successful");  
  
    } catch (AutoScalingException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DisableMetricsCollection](#) in *AWS SDK for Java 2.x API Reference*.

### Enable metrics collection for a group

The following code example shows how to enable CloudWatch metrics collection for an Auto Scaling group.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
    try {

        EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
            .autoScalingGroupName(groupName)
            .metrics("GroupMaxSize")
            .granularity("1Minute")
            .build();

        autoScalingClient.enableMetricsCollection(collectionRequest);
        System.out.println("The enable metrics collection operation was
successful");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [EnableMetricsCollection](#) in *AWS SDK for Java 2.x API Reference*.

## Get information about groups

The following code example shows how to get information about Auto Scaling groups.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String getAutoScaling( AutoScalingClient autoScalingClient, String
groupName) {
    try{
        String instanceId = "";
        DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest);
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        for (AutoScalingGroup group: groups) {
            System.out.println("The group name is " +
group.autoScalingGroupName());
            System.out.println("The group ARN is " + group.autoScalingGroupARN());

            List<Instance> instances = group.instances();
            for (Instance instance : instances) {
                instanceId = instance.instanceId();
            }
        }
        return instanceId;
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

```
}
```

- For API details, see [DescribeAutoScalingGroups](#) in *AWS SDK for Java 2.x API Reference*.

## Get information about instances

The following code example shows how to get information about Auto Scaling instances.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeAutoScalingInstance( AutoScalingClient autoScalingClient, String id) {
    try {
        DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest.builder()
            .instanceIds(id)
            .build();

        DescribeAutoScalingInstancesResponse response =
        autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
        List<AutoScalingInstanceDetails> instances =
        response.autoScalingInstances();
        for (AutoScalingInstanceDetails instance:instances ) {
            System.out.println("The instance lifecycle state is:
"+instance.lifecycleState());
        }
    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeAutoScalingInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Get information about scaling activities

The following code example shows how to get information about Auto Scaling activities.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeScalingActivities(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DescribeScalingActivitiesRequest scalingActivitiesRequest =
        DescribeScalingActivitiesRequest.builder()
            .autoScalingGroupName(groupName)
            .maxRecords(10)
```

```
        .build();

        DescribeScalingActivitiesResponse response =
autoScalingClient.describeScalingActivities(scalingActivitiesRequest);
        List<Activity> activities = response.activities();
        for (Activity activity: activities) {
            System.out.println("The activity Id is "+activity.activityId());
            System.out.println("The activity details are "+activity.details());
        }

    } catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [DescribeScalingActivities](#) in *AWS SDK for Java 2.x API Reference*.

## Set the desired capacity of a group

The following code example shows how to set the desired capacity of an Auto Scaling group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void setDesiredCapacity(AutoScalingClient autoScalingClient, String
groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
        .autoScalingGroupName(groupName)
        .desiredCapacity(2)
        .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SetDesiredCapacity](#) in *AWS SDK for Java 2.x API Reference*.

## Terminate an instance in a group

The following code example shows how to terminate an instance in an Auto Scaling group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void terminateInstanceInAutoScalingGroup(AutoScalingClient autoScalingClient, String instanceId){
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance "+instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [TerminateInstanceInAutoScalingGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Update a group

The following code example shows how to update the configuration for an Auto Scaling group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName, String launchTemplateName, String serviceLinkedRoleARN) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .serviceLinkedRoleARN(serviceLinkedRoleARN)
            .autoScalingGroupName(groupName)
            .launchTemplate(templateSpecification)
            .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter.waitUntilGroupInService(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("You successfully updated the auto scaling group
"+groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
```

- For API details, see [UpdateAutoScalingGroup in AWS SDK for Java 2.x API Reference](#).

## Scenarios

### Manage groups and instances

The following code example shows how to:

- Create an Amazon EC2 Auto Scaling group and configure it with a launch template and Availability Zones.
- Get information about the group and running instances.
- Enable Amazon CloudWatch metrics collection on the group.
- Update the desired capacity of the group and wait for an instance to start.
- Terminate an instance in the group.
- List scaling activities that occur in response to user requests and capacity changes.
- Get statistics for CloudWatch metrics that are collected during the example.
- Stop collecting metrics, terminate all instances, and delete the group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a launch template. For more information, see the following
 * topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
```

```
* 13. Deletes the Auto Scaling group.  
*/  
  
public class AutoScalingScenario {  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    public static void main(String[] args) throws InterruptedException {  
        final String usage = "\n" +  
            "Usage:\n" +  
            "  <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>\n" +  
            "\n" +  
            "Where:\n" +  
            "  groupName - The name of the Auto Scaling group.\n" +  
            "  launchTemplateName - The name of the launch template. \n" +  
            "  serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-  
linked role that the Auto Scaling group uses.\n" +  
            "  vpcZoneId - A subnet Id for a virtual private cloud (VPC) where  
instances in the Auto Scaling group can be created.\n" ;  
  
        if (args.length != 4) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String groupName = args[0];  
        String launchTemplateName = args[1];  
        String serviceLinkedRoleARN = args[2];  
        String vpcZoneId = args[3];  
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()  
            .region(Region.US_EAST_1)  
            .credentialsProvider(ProfileCredentialsProvider.create())  
            .build();  
  
        System.out.println(DASHES);  
        System.out.println("Welcome to the Amazon EC2 Auto Scaling example scenario.");  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("1. Create an Auto Scaling group named "+groupName);  
        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,  
            serviceLinkedRoleARN, vpcZoneId);  
        System.out.println("Wait 1 min for the resources, including the instance.  
Otherwise, an empty instance Id is returned");  
        Thread.sleep(60000);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("2. Get Auto Scale group Id value");  
        String instanceId = getSpecificAutoScalingGroups(autoScalingClient, groupName);  
        if (instanceId.compareTo("") ==0) {  
            System.out.println("Error - no instance Id value");  
            System.exit(1);  
        } else {  
            System.out.println("The instance Id value is "+instanceId);  
        }  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("3. Describe Auto Scaling with the Id value "+instanceId);  
        describeAutoScalingInstance( autoScalingClient, instanceId);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("4. Enable metrics collection "+instanceId);  
        enableMetricsCollection(autoScalingClient, groupName);  
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("5. Update an Auto Scaling group to update max size to 3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
        serviceLinkedRoleARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println("Wait 1 min for the resources, including the instance.
Otherwise, an empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred for the
group");
        describeScalingActivities(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group");
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed." );
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static void describeScalingActivities(AutoScalingClient autoScalingClient,
String groupName) {
    try {
        DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
            .autoScalingGroupName(groupName)
            .maxRecords(10)
            .build();
    }
}
```

```
        DescribeScalingActivitiesResponse response =
autoScalingClient.describeScalingActivities(scalingActivitiesRequest);
    List<Activity> activities = response.activities();
    for (Activity activity: activities) {
        System.out.println("The activity Id is "+activity.activityId());
        System.out.println("The activity details are "+activity.details());
    }

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void setDesiredCapacity(AutoScalingClient autoScalingClient, String
groupName) {
    try {
        SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
            .autoScalingGroupName(groupName)
            .desiredCapacity(2)
            .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
                                         String groupName,
                                         String launchTemplateName,
                                         String serviceLinkedRoleARN,
                                         String vpcZoneId) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .availabilityZones("us-east-1a")
            .launchTemplate(templateSpecification)
            .maxSize(1)
            .minSize(1)
            .vpcZoneIdentifier(vpcZoneId)
            .serviceLinkedRoleARN(serviceLinkedRoleARN)
            .build();

        autoScalingClient.createAutoScalingGroup(request);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter.waitUntilGroupExists(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");
    }
}
```

```
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingInstance( AutoScalingClient autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest =
DescribeAutoScalingInstancesRequest.builder()
                .instanceIds(id)
                .build();

            DescribeAutoScalingInstancesResponse response =
autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance:instances ) {
                System.out.println("The instance lifecycle state is:
"+instance.lifecycleState());
            }
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingGroups(AutoScalingClient autoScalingClient, String groupName) {
        try {
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .maxRecords(10)
                .build();

            DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group: groups) {
                System.out.println("**** The service to use for the health checks: "+
group.healthCheckType());
            }
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSpecificAutoScalingGroups(AutoScalingClient autoScalingClient, String groupName) {
        try{
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

            DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group: groups) {
```

```
        System.out.println("The group name is " +
group.autoScalingGroupName());
        System.out.println("The group ARN is " + group.autoScalingGroupARN());
List<Instance> instances = group.instances();

        for (Instance instance : instances) {
            instanceId = instance.instanceId();
            System.out.println("The instance id is " + instanceId);
            System.out.println("The lifecycle state is "
+instance.lifecycleState());
        }
    }

    return instanceId ;
} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "" ;
}

public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
try {

    EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
        .autoScalingGroupName(groupName)
        .metrics("GroupMaxSize")
        .granularity("1Minute")
        .build();

    autoScalingClient.enableMetricsCollection(collectionRequest);
    System.out.println("The enable metrics collection operation was
successful");

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
try {
    DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
        .autoScalingGroupName(groupName)
        .metrics("GroupMaxSize")
        .build();

autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
    System.out.println("The disable metrics collection operation was
successful");

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void describeAccountLimits(AutoScalingClient autoScalingClient) {
try {
    DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
```

```
        System.out.println("The max number of auto scaling groups is
"+response.maxNumberOfAutoScalingGroups());
        System.out.println("The current number of auto scaling groups is
"+response.numberOfAutoScalingGroups());

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName, String launchTemplateName, String serviceLinkedRoleARN) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
            .maxSize(3)
            .serviceLinkedRoleARN(serviceLinkedRoleARN)
            .autoScalingGroupName(groupName)
            .launchTemplate(templateSpecification)
            .build();

        autoScalingClient.updateAutoScalingGroup(groupRequest);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter.waitUntilGroupInService(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("You successfully updated the auto scaling group
"+groupName);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId){
    try {
        TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

        autoScalingClient.terminateInstanceInAutoScalingGroup(request);
        System.out.println("You have terminated instance "+instanceId);

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
```

```
try {
    DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
    .autoScalingGroupName(groupName)
    .forceDelete(true)
    .build() ;

    autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest) ;
    System.out.println("You successfully deleted "+groupName);

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [CreateAutoScalingGroup](#)
  - [DeleteAutoScalingGroup](#)
  - [DescribeAutoScalingGroups](#)
  - [DescribeAutoScalingInstances](#)
  - [DescribeScalingActivities](#)
  - [DisableMetricsCollection](#)
  - [EnableMetricsCollection](#)
  - [SetDesiredCapacity](#)
  - [TerminateInstanceInAutoScalingGroup](#)
  - [UpdateAutoScalingGroup](#)

## EventBridge examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon EventBridge.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3273\)](#)

## Actions

### Create a scheduled rule

The following code example shows how to create an Amazon EventBridge scheduled rule.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createEBRule(EventBridgeClient eventBrClient, String ruleName, String cronExpression) {

    try {
        PutRuleRequest ruleRequest = PutRuleRequest.builder()
            .name(ruleName)
            .eventBusName("default")
            .scheduleExpression(cronExpression)
            .state("ENABLED")
            .description("A test rule that runs on a schedule created by the
Java API")
            .build();

        PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
        System.out.println("The ARN of the new rule is "+
ruleResponse.ruleArn());

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutRule](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a scheduled rule

The following code example shows how to delete an Amazon EventBridge scheduled rule.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteEBRule(EventBridgeClient eventBrClient, String ruleName) {

    try {
        DisableRuleRequest disableRuleRequest = DisableRuleRequest.builder()
            .name(ruleName)
            .eventBusName("default")
            .build();

        eventBrClient.disableRule(disableRuleRequest);
        DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
            .name(ruleName)
            .eventBusName("default")
            .build();

        eventBrClient.deleteRule(ruleRequest);
        System.out.println("Rule "+ruleName+" was successfully deleted!");

    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteRule](#) in *AWS SDK for Java 2.x API Reference*.

## Send events

The following code example shows how to send Amazon EventBridge events.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void putEBEvents(EventBridgeClient eventBrClient, String resourceArn,
String resourceArn2 ) {

    try {
        // Populate a List with the resource ARN values.
        List<String> resources = new ArrayList<>();
        resources.add(resourceArn);
        resources.add(resourceArn2);

        PutEventsRequestEntry reqEntry = PutEventsRequestEntry.builder()
            .resources(resources)
            .source("com.mycompany.myapp")
            .detailType("myDetailType")
            .detail("{ \"key1\": \"value1\", \"key2\": \"value2\" }")
            .build();

        PutEventsRequest eventsRequest = PutEventsRequest.builder()
            .entries(reqEntry)
            .build();

        PutEventsResponse result = eventBrClient.putEvents(eventsRequest);
        for (PutEventsResultEntry resultEntry : result.entries()) {
            if (resultEntry.eventId() != null) {
                System.out.println("Event Id: " + resultEntry.eventId());
            } else {
                System.out.println("Injection failed with Error Code: " +
resultEntry.errorCode());
            }
        }
    } catch (EventBridgeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutEvents](#) in *AWS SDK for Java 2.x API Reference*.

## AWS Glue examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS Glue.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3276\)](#)
- [Scenarios \(p. 3279\)](#)

## Actions

### Create a crawler

The following code example shows how to create an AWS Glue crawler.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createGlueCrawler(GlueClient glueClient,
                                      String iam,
                                      String s3Path,
                                      String cron,
                                      String dbName,
                                      String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        // Add the S3Target to a list.
        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);

        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
            .schedule(cron)
            .build();

        glueClient.createCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully created");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateCrawler](#) in *AWS SDK for Java 2.x API Reference*.

### Get a crawler

The following code example shows how to get an AWS Glue crawler.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getSpecificCrawler(GlueClient glueClient, String crawlerName) {  
  
    try {  
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()  
            .name(crawlerName)  
            .build();  
  
        GetCrawlerResponse response = glueClient.getcrawler(crawlerRequest);  
        Instant createDate = response.crawler().creationTime();  
  
        // Convert the Instant to readable date  
        DateTimeFormatter formatter =  
            DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)  
                .withLocale(Locale.US)  
                .withZone(ZoneId.systemDefault());  
  
        formatter.format(createDate);  
        System.out.println("The create date of the Crawler is " + createDate);  
  
    } catch (GlueException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [GetCrawler](#) in *AWS SDK for Java 2.x API Reference*.

## Get a database from the Data Catalog

The following code example shows how to get a database from the AWS Glue Data Catalog.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getSpecificDatabase(GlueClient glueClient, String databaseName) {  
  
    try {  
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()  
            .name(databaseName)  
            .build();  
  
        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);  
        Instant createDate = response.database().createTime();  
  
        // Convert the Instant to readable date.  
        DateTimeFormatter formatter =  
            DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)  
                .withLocale(Locale.US)  
                .withZone(ZoneId.systemDefault());  
  
        formatter.format(createDate);  
    }
```

```
        System.out.println("The create date of the database is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [GetDatabase](#) in *AWS SDK for Java 2.x API Reference*.

## Get tables from a database

The following code example shows how to get tables from a database in the AWS Glue Data Catalog.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getGlueTable(GlueClient glueClient, String dbName, String
tableName ) {

    try {
        GetTableRequest tableRequest = GetTableRequest.builder()
            .databaseName(dbName)
            .name(tableName)
            .build();

        GetTableResponse tableResponse = glueClient.getTable(tableRequest);
        Instant createDate = tableResponse.table().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )
            .withLocale( Locale.US )
            .withZone( ZoneId.systemDefault() );

        formatter.format( createDate );
        System.out.println("The create date of the table is " + createDate );

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [GetTables](#) in *AWS SDK for Java 2.x API Reference*.

## Start a crawler

The following code example shows how to start an AWS Glue crawler.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void startSpecificCrawler(GlueClient glueClient, String crawlerName) {
    try {
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.startCrawler(crawlerRequest);
    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [StartCrawler](#) in *AWS SDK for Java 2.x API Reference*.

## Scenarios

### Get started running crawlers and jobs

The following code example shows how to:

- Create and run a crawler that crawls a public Amazon Simple Storage Service (Amazon S3) bucket and generates a metadata database that describes the CSV-formatted data it finds.
- List information about databases and tables in your AWS Glue Data Catalog.
- Create and run a job that extracts CSV data from the source Amazon S3 bucket, transforms it by removing and renaming fields, and loads JSON-formatted output into another Amazon S3 bucket.
- List information about job runs and view some of the transformed data.
- Delete all resources created by the demo.

For more information, see [Tutorial: Getting started with AWS Glue Studio](#).

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
 * Before running this Java V2 code example, set up your development environment,  
 * including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * To set up the resources, see this documentation topic:  
 *  
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html  
 *  
 * This example performs the following tasks:  
 *  
 * 1. Create a database.  
 * 2. Create a crawler.  
 * 3. Get a crawler.
```

```
* 4. Start a crawler.  
* 5. Get a database.  
* 6. Get tables.  
* 7. Create a job.  
* 8. Start a job run.  
* 9. List all jobs.  
* 10. Get job runs.  
* 11. Delete a job.  
* 12. Delete a database.  
* 13. Delete a crawler.  
*/  
  
public class GlueScenario {  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    public static void main(String[] args) throws InterruptedException {  
  
        final String usage = "\n" +  
            "Usage:\n" +  
            "    <iام> <s3Path> <cron> <dbName> <crawlerName> <jobName> \n\n" +  
            "Where:\n" +  
            "    iam - The ARN of the IAM role that has AWS Glue and S3 permissions.  
\n" +  
            "    s3Path - The Amazon Simple Storage Service (Amazon S3) target that  
contains data (for example, CSV data).\n" +  
            "    cron - A cron expression used to specify the schedule (i.e., cron(15  
12 * * ? *).\n" +  
            "    dbName - The database name. \n" +  
            "    crawlerName - The name of the crawler. \n" +  
            "    jobName - The name you assign to this job definition." +  
            "    scriptLocation - The Amazon S3 path to a script that runs a job." +  
            "    locationUri - The location of the database" ;  
  
        if (args.length != 8) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String iam = args[0];  
        String s3Path = args[1];  
        String cron = args[2];  
        String dbName = args[3];  
        String crawlerName = args[4];  
        String jobName = args[5];  
        String scriptLocation = args[6];  
        String locationUri = args[7];  
  
        Region region = Region.US_EAST_1;  
        GlueClient glueClient = GlueClient.builder()  
            .region(region)  
            .credentialsProvider(ProfileCredentialsProvider.create())  
            .build();  
        System.out.println(DASHES);  
        System.out.println("Welcome to the AWS Glue scenario.");  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("1. Create a database.");  
        createDatabase(glueClient, dbName, locationUri);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("2. Create a crawler.");  
        createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);
```

```
System.out.println("3. Get a crawler.");
getSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Start a crawler.");
startSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a database.");
getSpecificDatabase(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Get tables.");
getGlueTables(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create a job.");
createJob(glueClient, jobName, iam, scriptLocation);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Start a Job run.");
startJob(glueClient, jobName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. List all jobs.");
getAllJobs(glueClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get job runs.");
getJobRuns(glueClient, jobName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Delete a job.");
deleteJob(glueClient, jobName);
System.out.println("*** Wait 5 MIN for the "+crawlerName +" to stop");
TimeUnit.MINUTES.sleep(5);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Delete a database.");
deleteDatabase(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Delete a crawler.");
deleteSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Successfully completed the AWS Glue Scenario");
System.out.println(DASHES);
}

public static void createDatabase(GlueClient glueClient, String dbName, String
locationUri ) {

try {
    DatabaseInput input = DatabaseInput.builder()
```

```
.description("Built with the AWS SDK for Java V2")
.name(dbName)
.locationUri(locationUri)
.build();

CreateDatabaseRequest request = CreateDatabaseRequest.builder()
.databaseInput(input)
.build();

glueClient.createDatabase(request);
System.out.println("The database was successfully created");

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

public static void createGlueCrawler(GlueClient glueClient,
        String iam,
        String s3Path,
        String cron,
        String dbName,
        String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);
        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
            .schedule(cron)
            .build();

        glueClient.createCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully created");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificCrawler(GlueClient glueClient, String crawlerName) {

    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        GetCrawlerResponse response = glueClient.getcrawler(crawlerRequest);
        Instant createDate = response.crawler().creationTime();
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT )
            .withLocale( Locale.US)
```

```
.withZone( ZoneId.systemDefault() );

formatter.format( createDate );
System.out.println("The create date of the Crawler is " + createDate );

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void startSpecificCrawler(GlueClient glueClient, String crawlerName)
{

    try {
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.startCrawler(crawlerRequest);
        System.out.println(crawlerName +" was successfully started!");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificDatabase(GlueClient glueClient, String databaseName)
{

    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )
            .withLocale( Locale.US )
            .withZone( ZoneId.systemDefault() );

        formatter.format( createDate );
        System.out.println("The create date of the database is " + createDate );

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getGlueTables(GlueClient glueClient, String dbName){

    try {
        GetTablesRequest tableRequest = GetTablesRequest.builder()
            .databaseName(dbName)
            .build();

        GetTablesResponse response = glueClient.getTables(tableRequest);
        List<Table> tables = response.tableList();
        for (Table table: tables) {
            System.out.println("Table name is: "+table.name());
        }
    }
}
```

```
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void startJob(GlueClient glueClient, String jobName) {
        try {
            StartJobRunRequest runRequest = StartJobRunRequest.builder()
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .jobName(jobName)
                .build();

            StartJobRunResponse response = glueClient.startJobRun(runRequest);
            System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {
        try {
            JobCommand command = JobCommand.builder()
                .pythonVersion("3")
                .name("MyJob1")
                .scriptLocation(scriptLocation)
                .build();

            CreateJobRequest jobRequest = CreateJobRequest.builder()
                .description("A Job created by using the AWS SDK for Java V2")
                .glueVersion("2.0")
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .name(jobName)
                .role(iam)
                .command(command)
                .build();

            glueClient.createJob(jobRequest);
            System.out.println(jobName +" was successfully created.");
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getAllJobs(GlueClient glueClient) {
        try {
            GetJobsRequest jobsRequest = GetJobsRequest.builder()
                .maxResults(10)
                .build();

            GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job: jobs) {
                System.out.println("Job name is : "+job.name());
            }
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
        System.out.println("The job worker type is :  
"+job.workerType().name());  
    }  
  
    } catch (GlueException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}  
  
public static void getJobRuns(GlueClient glueClient, String jobName) {  
  
    try {  
        GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()  
            .jobName(jobName)  
            .maxResults(20)  
            .build();  
  
        GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);  
        List<JobRun> jobRuns = response.jobRuns();  
        for (JobRun jobRun: jobRuns) {  
            System.out.println("Job run state is "+jobRun.jobRunState().name());  
            System.out.println("Job run Id is "+jobRun.id());  
            System.out.println("The Glue version is "+jobRun.glueVersion());  
        }  
  
    } catch (GlueException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}  
  
public static void deleteJob(GlueClient glueClient, String jobName) {  
  
    try {  
        DeleteJobRequest jobRequest = DeleteJobRequest.builder()  
            .jobName(jobName)  
            .build();  
  
        glueClient.deleteJob(jobRequest);  
        System.out.println(jobName +" was successfully deleted");  
  
    } catch (GlueException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}  
  
public static void deleteDatabase(GlueClient glueClient, String databaseName) {  
  
    try {  
        DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()  
            .name(databaseName)  
            .build();  
  
        glueClient.deleteDatabase(request);  
        System.out.println(databaseName +" was successfully deleted");  
  
    } catch (GlueException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}  
  
public static void deleteSpecificCrawler(GlueClient glueClient, String crawlerName)  
{
```

```
try {
    DeleteCrawlerRequest deleteCrawlerRequest = DeleteCrawlerRequest.builder()
        .name(crawlerName)
        .build();

    glueClient.deleteCrawler(deleteCrawlerRequest);
    System.out.println(crawlerName +" was deleted");

} catch (GlueException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.

- [CreateCrawler](#)
- [CreateJob](#)
- [DeleteCrawler](#)
- [DeleteDatabase](#)
- [DeleteJob](#)
- [DeleteTable](#)
- [GetCrawler](#)
- [GetDatabase](#)
- [GetDatabases](#)
- [GetJob](#)
- [GetJobRun](#)
- [GetJobRuns](#)
- [GetTables](#)
- [ListJobs](#)
- [StartCrawler](#)
- [StartJobRun](#)

## IAM examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS Identity and Access Management.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3286\)](#)
- [Scenarios \(p. 3296\)](#)

## Actions

### Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String policyArn ) {

    try {
        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
        .roleName(roleName)
        .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        // Ensure that the policy is not attached to this role
        String polArn = "";
        for (AttachedPolicy policy: attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn)==0) {
                System.out.println(roleName + " policy is already attached to this
role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest = AttachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(policyArn)
        .build();

        iam.attachRolePolicy(attachRequest);

        System.out.println("Successfully attached policy " + policyArn +
                           " to role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Java 2.x API Reference*.

## Create a policy

The following code example shows how to create an IAM policy.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createIAMPolicy(IamClient iam, String policyName ) {
```

```
try {
    // Create an IamWaiter object.
    IamWaiter iamWaiter = iam.waiter();

    CreatePolicyRequest request = CreatePolicyRequest.builder()
        .policyName(policyName)
        .policyDocument(PolicyDocument)
        .build();

    CreatePolicyResponse response = iam.createPolicy(request);

    // Wait until the policy is created.
    GetPolicyRequest polRequest = GetPolicyRequest.builder()
        .policyArn(response.policy().arn())
        .build();

    WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
    iamWaiter.waitUntilPolicyExists(polRequest);
    waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
    return response.policy().arn();

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Java 2.x API Reference*.

## Create a role

The following code example shows how to create an IAM role.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createIAMRole(IamClient iam, String rolename, String
fileLocation ) throws Exception {

    try {
        JSONObject json0bject = (JSONObject) readJsonSimpleDemo(fileLocation);
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(json0bject.toJSONString())
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is "+response.role().arn());

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

```
public static Object readJsonSimpleDemo(String filename) throws Exception {
    FileReader reader = new FileReader(filename);
    JSONParser jsonParser = new JSONParser();
    return jsonParser.parse(reader);
}
```

- For API details, see [CreateRole](#) in *AWS SDK for Java 2.x API Reference*.

## Create a user

The following code example shows how to create an IAM user.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createIAMUser(IamClient iam, String username ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        CreateUserResponse response = iam.createUser(request);

        // Wait until the user is created
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
        iamWaiter.waitUntilUserExists(userRequest);
        waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return response.user().userName();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateUser](#) in *AWS SDK for Java 2.x API Reference*.

## Create an access key

The following code example shows how to create an IAM access key.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createIAMAccessKey(IamClient iam, String user) {  
  
    try {  
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()  
            .userName(user)  
            .build();  
  
        CreateAccessKeyResponse response = iam.createAccessKey(request);  
        return response.accessKey().accessKeyId();  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for Java 2.x API Reference*.

## Create an alias for an account

The following code example shows how to create an alias for an IAM account.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createIAMAccountAlias(IamClient iam, String alias) {  
  
    try {  
        CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()  
            .accountAlias(alias)  
            .build();  
  
        iam.createAccountAlias(request);  
        System.out.println("Successfully created account alias: " + alias);  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [CreateAccountAlias](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a policy

The following code example shows how to delete an IAM policy.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteIAMPolicy(IamClient iam, String policyARN) {  
  
    try {  
        DeletePolicyRequest request = DeletePolicyRequest.builder()  
            .policyArn(policyARN)  
            .build();  
  
        iam.deletePolicy(request);  
        System.out.println("Successfully deleted the policy");  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    System.out.println("Done");  
}
```

- For API details, see [DeletePolicy](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a user

The following code example shows how to delete an IAM user.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteIAMUser(IamClient iam, String userName) {  
  
    try {  
        DeleteUserRequest request = DeleteUserRequest.builder()  
            .userName(userName)  
            .build();  
  
        iam.deleteUser(request);  
        System.out.println("Successfully deleted IAM user " + userName);  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteUser](#) in *AWS SDK for Java 2.x API Reference*.

## Delete an access key

The following code example shows how to delete an IAM access key.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteKey(IamClient iam ,String username, String accessKey ) {  
  
    try {  
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()  
            .accessKeyId(accessKey)  
            .userName(username)  
            .build();  
  
        iam.deleteAccessKey(request);  
        System.out.println("Successfully deleted access key " + accessKey +  
            " from user " + username);  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for Java 2.x API Reference*.

## Delete an account alias

The following code example shows how to delete an IAM account alias.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteIAMAccountAlias(IamClient iam, String alias ) {  
  
    try {  
        DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()  
            .accountAlias(alias)  
            .build();  
  
        iam.deleteAccountAlias(request);  
        System.out.println("Successfully deleted account alias " + alias);  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    System.out.println("Done");  
}
```

- For API details, see [DeleteAccountAlias](#) in *AWS SDK for Java 2.x API Reference*.

## Detach a policy from a role

The following code example shows how to detach an IAM policy from a role.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detachPolicy(IamClient iam, String roleName, String policyArn ) {
    try {
        DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.detachRolePolicy(request);
        System.out.println("Successfully detached policy " + policyArn +
            " from role " + roleName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DetachRolePolicy](#) in [AWS SDK for Java 2.x API Reference](#).

## List a user's access keys

The following code example shows how to list a user's IAM access keys.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listKeys( IamClient iam, String userName ){

    try {
        boolean done = false;
        String newMarker = null;

        while (!done) {
            ListAccessKeysResponse response;

            if(newMarker == null) {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .build();

                response = iam.listAccessKeys(request);
            } else {
                ListAccessKeysRequest request = ListAccessKeysRequest.builder()
                    .userName(userName)
                    .marker(newMarker)
                    .build();

                response = iam.listAccessKeys(request);
            }

            for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                System.out.format("Retrieved access key %s",
                    metadata.accessKeyId());
            }
        }
    }
}
```

```
        if (!response.isTruncated()) {
            done = true;
        } else {
            newMarker = response.marker();
        }
    }

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [ListAccessKeys](#) in *AWS SDK for Java 2.x API Reference*.

## List account aliases

The following code example shows how to list IAM account aliases.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAliases(IamClient iam) {

    try {
        ListAccountAliasesResponse response = iam.listAccountAliases();
        for (String alias : response.accountAliases()) {
            System.out.printf("Retrieved account alias %s", alias);
        }
    }

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [ListAccountAliases](#) in *AWS SDK for Java 2.x API Reference*.

## List users

The following code example shows how to list IAM users.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllUsers(IamClient iam ) {

    try {

        boolean done = false;
```

```
String newMarker = null;

while(!done) {
    ListUsersResponse response;
    if (newMarker == null) {
        ListUsersRequest request = ListUsersRequest.builder().build();
        response = iam.listUsers(request);
    } else {
        ListUsersRequest request = ListUsersRequest.builder()
            .marker(newMarker)
            .build();

        response = iam.listUsers(request);
    }

    for(User user : response.users()) {
        System.out.format("\n Retrieved user %s", user.userName());
        AttachedPermissionsBoundary permissionsBoundary =
        user.permissionsBoundary();
        if (permissionsBoundary != null)
            System.out.format("\n Permissions boundary details %s",
permissionsBoundary.permissionsBoundaryTypeAsString());
    }

    if(!response.isTruncated()) {
        done = true;
    } else {
        newMarker = response.marker();
    }
}

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Java 2.x API Reference*.

## Update a user

The following code example shows how to update an IAM user.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void updateIAMUser(IamClient iam, String curName, String newName ) {

    try {
        UpdateUserRequest request = UpdateUserRequest.builder()
            .userName(curName)
            .newUserName(newName)
            .build();

        iam.updateUser(request);
        System.out.printf("Successfully updated user to username %s", newName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
```

- For API details, see [UpdateUser](#) in *AWS SDK for Java 2.x API Reference*.

## Update an access key

The following code example shows how to update an IAM access key.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void updateKey(IamClient iam, String username, String accessId,
String status) {
    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
            statusType = StatusType.INACTIVE;
        } else {
            statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
        }

        UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
            .accessKeyId(accessId)
            .userName(username)
            .status(statusType)
            .build();

        iam.updateAccessKey(request);
        System.out.printf("Successfully updated the status of access key %s to" +
            "status %s for user %s", accessId, status, username);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [UpdateAccessKey](#) in *AWS SDK for Java 2.x API Reference*.

## Scenarios

### Create a user and assume a role

The following code example shows how to:

- Create a user who has no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list Amazon S3 buckets using temporary credentials.
- Delete the policy, role, and user.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap IAM user actions.

```
/*
 To run this Java V2 code example, set up your development environment, including your
 credentials.

 For information, see this documentation topic:

 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

 This example performs these operations:

 1. Creates a user that has no permissions.
 2. Creates a role and policy that grants Amazon S3 permissions.
 3. Creates a role.
 4. Grants the user permissions.
 5. Gets temporary credentials by assuming the role. Creates an Amazon S3 Service
 client object with the temporary credentials.
 6. Deletes the resources.
 */

public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static final String PolicyDocument =
        "{" +
            "  \"Version\": \"2012-10-17\", " +
            "  \"Statement\": [" +
            "    {" +
            "      \"Effect\": \"Allow\", " +
            "      \"Action\": [\"" +
            "        \"s3:*\" " +
            "      ], " +
            "      \"Resource\": \"*\" " +
            "    }" +
            "  ]" +
        "}";
}

public static void main(String[] args) throws Exception {
    final String usage = "\n" +
        "Usage:\n" +
        "      <username> <policyName> <roleName> <roleSessionName> <fileLocation>
<bucketName> \n\n" +
        "Where:\n" +
        "      username - The name of the IAM user to create. \n\n" +
        "      policyName - The name of the policy to create. \n\n" +
        "      roleName - The name of the role to create. \n\n" +
        "      roleSessionName - The name of the session required for the assumeRole
operation. \n\n" +
        "      fileLocation - The file location to the JSON required to create the
role (see Readme). \n\n" +
        "      bucketName - The name of the Amazon S3 bucket from which objects are
read. \n\n" ;

    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }
}
```

```
String userName = args[0];
String policyName = args[1];
String roleName = args[2];
String roleSessionName = args[3];
String fileLocation = args[4];
String bucketName = args[5];

Region region = Region.AWS_GLOBAL;
IamClient iam = IamClient.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the AWS IAM example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(" 1. Create the IAM user.");
Boolean createUser = createIAMUser(iam, userName);
System.out.println(DASHES);

if (createUser) {
    System.out.println(userName + " was successfully created.");

    System.out.println(DASHES);
    System.out.println("2. Creates a policy.");
    String polArn = createIAMPolicy(iam, policyName);
    System.out.println("The policy " + polArn + " was successfully created.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Creates a role.");
    String roleArn = createIAMRole(iam, roleName, fileLocation);
    System.out.println(roleArn + " was successfully created.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Grants the user permissions.");
    attachIAMRolePolicy(iam, roleName, polArn);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("*** Wait for 1 MIN so the resource is available");
    TimeUnit.MINUTES.sleep(1);
    System.out.println("5. Gets temporary credentials by assuming the role.");
    System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
    assumeGivenRole(roleArn, roleSessionName, bucketName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6 Getting ready to delete the AWS resources");
    deleteRole(iam, roleName, polArn);
    deleteIAMUser(iam, userName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("This IAM Scenario has successfully completed");
    System.out.println(DASHES);
} else {
    System.out.println(userName +" was not successfully created.");
}
}

public static Boolean createIAMUser(IamClient iam, String username ) {
```

```
try {
    // Create an IamWaiter object
    IamWaiter iamWaiter = iam.waiter();
    CreateUserRequest request = CreateUserRequest.builder()
        .userName(username)
        .build();

    // Wait until the user is created.
    CreateUserResponse response = iam.createUser(request);
    GetUserRequest userRequest = GetUserRequest.builder()
        .userName(response.user().userName())
        .build();

    WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
    waitUntilUserExists.matched().response().ifPresent(System.out::println);
    return true;

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return false;
}

public static String createIAMRole(IamClient iam, String rolename, String
fileLocation ) throws Exception {

try {
    JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
    CreateRoleRequest request = CreateRoleRequest.builder()
        .roleName(rolename)
        .assumeRolePolicyDocument(jsonObject.toJSONString())
        .description("Created using the AWS SDK for Java")
        .build();

    CreateRoleResponse response = iam.createRole(request);
    System.out.println("The ARN of the role is "+response.role().arn());
    return response.role().arn();

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static String createIAMPolicy(IamClient iam, String policyName ) {

try {
    // Create an IamWaiter object.
    IamWaiter iamWaiter = iam.waiter();
    CreatePolicyRequest request = CreatePolicyRequest.builder()
        .policyName(policyName)
        .policyDocument(PolicyDocument).build();

    CreatePolicyResponse response = iam.createPolicy(request);

    // Wait until the policy is created.
    GetPolicyRequest polRequest = GetPolicyRequest.builder()
        .policyArn(response.policy().arn())
        .build();

    WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);
}
```

```
        waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

    try {
        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
        .roleName(roleName)
        .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        String polArn;
        for (AttachedPolicy policy: attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn)==0) {
                System.out.println(roleName + " policy is already attached to this
role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest = AttachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(policyArn)
        .build();

        iam.attachRolePolicy(attachRequest);
        System.out.println("Successfully attached policy " + policyArn + " to role
" + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Invoke an Amazon S3 operation using the Assumed Role.
public static void assumeGivenRole(String roleArn, String roleSessionName, String
bucketName) {

    StsClient stsClient = StsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
        .roleArn(roleArn)
        .roleSessionName(roleSessionName)
        .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials myCreds = roleResponse.credentials();
        String key = myCreds.accessKeyId();
        String secKey = myCreds.secretAccessKey();
    }
}
```

```
String secToken = myCreds.sessionToken();

// List all objects in an Amazon S3 bucket using the temp creds.
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()

.credentialsProvider(StaticCredentialsProvider.create(AwsSessionCredentials.create(key,
secKey, secToken)))
.region(region)
.build();

System.out.println("Created a S3Client using temp credentials.");
System.out.println("Listing objects in "+bucketName);
ListObjectsRequest listObjects = ListObjectsRequest.builder()
.bucket(bucketName)
.build();

ListObjectsResponse res = s3.listObjects(listObjects);
List<S3Object> objects = res.contents();
for (S3Object myValue : objects) {
    System.out.println("The name of the key is " + myValue.key());
    System.out.println("The owner is " + myValue.owner());
}

} catch (StsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void deleteRole(IamClient iam, String roleName, String polArn) {

try {
    // First the policy needs to be detached.
    DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
    .policyArn(polArn)
    .roleName(roleName)
    .build();

    iam.detachRolePolicy(rolePolicyRequest);

    // Delete the policy.
    DeletePolicyRequest request = DeletePolicyRequest.builder()
    .policyArn(polArn)
    .build();

    iam.deletePolicy(request);
    System.out.println("*** Successfully deleted "+polArn);

    // Delete the role.
    DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
    .roleName(roleName)
    .build();

    iam.deleteRole(roleRequest);
    System.out.println("*** Successfully deleted " +roleName);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void deleteIAMUser(IamClient iam, String userName) {
```

```
try {
    DeleteUserRequest request = DeleteUserRequest.builder()
        .userName(userName)
        .build();

    iam.deleteUser(request);
    System.out.println("*** Successfully deleted " + userName);

} catch (IamException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

public static Object readJsonSimpleDemo(String filename) throws Exception {
    FileReader reader = new FileReader(filename);
    JSONParser jsonParser = new JSONParser();
    return jsonParser.parse(reader);
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## AWS KMS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS Key Management Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3302\)](#)

## Actions

### Create a grant for a key

The following code example shows how to create a grant for a KMS key.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createGrant(KmsClient kmsClient, String keyId, String
granteePrincipal, String operation) {

    try {
        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .granteePrincipal(granteePrincipal)
            .operationsWithStrings(operation)
            .build();

        CreateGrantResponse response = kmsClient.createGrant(grantRequest);
        return response.grantId();

    }catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateGrant](#) in *AWS SDK for Java 2.x API Reference*.

## Create a key

The following code example shows how to create an AWS KMS key.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createKey(KmsClient kmsClient, String keyDesc) {

    try {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
            .keyUsage("ENCRYPT_DECRYPT")
            .build();

        CreateKeyResponse result = kmsClient.createKey(keyRequest);
        System.out.printf("Created a customer key with id \"%s\"\n",
result.keyMetadata().arn());
        return result.keyMetadata().keyId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateKey](#) in *AWS SDK for Java 2.x API Reference*.

## Create an alias for a key

The following code example shows how to create an alias for a KMS key key.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {

    try {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        kmsClient.createAlias(aliasRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateAlias](#) in *AWS SDK for Java 2.x API Reference*.

## Decrypt ciphertext

The following code example shows how to decrypt ciphertext that was encrypted by a KMS key.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void decryptData(KmsClient kmsClient, SdkBytes encryptedData, String
keyId) {

    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        decryptResponse.plaintext();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [Decrypt](#) in *AWS SDK for Java 2.x API Reference*.

## Describe a key

The following code example shows how to describe a KMS key.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeSpecifcKey(KmsClient kmsClient, String keyId ){  
    try {  
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()  
            .keyId(keyId)  
            .build();  
  
        DescribeKeyResponse response = kmsClient.describeKey(keyRequest);  
        System.out.println("The key description is  
"+response.keyMetadata().description());  
        System.out.println("The key ARN is "+response.keyMetadata().arn());  
  
    } catch (KmsException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DescribeKey](#) in *AWS SDK for Java 2.x API Reference*.

## Disable a key

The following code example shows how to disable a KMS key.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void disableKey( KmsClient kmsClient, String keyId ) {  
    try {  
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()  
            .keyId(keyId)  
            .build();  
  
        kmsClient.disableKey(keyRequest);  
  
    } catch (KmsException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DisableKey](#) in *AWS SDK for Java 2.x API Reference*.

## Enable a key

The following code example shows how to enable a KMS key.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void enableKey(KmsClient kmsClient, String keyId) {  
  
    try {  
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()  
            .keyId(keyId)  
            .build();  
  
        kmsClient.enableKey(enableKeyRequest);  
  
    } catch (KmsException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [EnableKey](#) in *AWS SDK for Java 2.x API Reference*.

## Encrypt text using a key

The following code example shows how to encrypt text using a KMS key.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static SdkBytes encryptData(KmsClient kmsClient, String keyId) {  
  
    try {  
        SdkBytes myBytes = SdkBytes.fromByteArray(new byte[]{1, 2, 3, 4, 5, 6, 7,  
8, 9, 0});  
  
        EncryptRequest encryptRequest = EncryptRequest.builder()  
            .keyId(keyId)  
            .plaintext(myBytes)  
            .build();  
  
        EncryptResponse response = kmsClient.encrypt(encryptRequest);  
        String algorithm = response.encryptionAlgorithm().toString();  
        System.out.println("The encryption algorithm is " + algorithm);  
  
        // Get the encrypted data.  
        SdkBytes encryptedData = response.ciphertextBlob();  
    } catch (KmsException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

```
        return encryptedData;
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return null;
}
```

- For API details, see [Encrypt in AWS SDK for Java 2.x API Reference](#).

## List aliases for a key

The following code example shows how to list aliases for a KMS key.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllAliases( KmsClient kmsClient ) {

    try {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesResponse aliasesResponse =
        kmsClient.listAliases(aliasesRequest) ;
        List<AliasListEntry> aliases = aliasesResponse_aliases();
        for (AliasListEntry alias: aliases) {
            System.out.println("The alias name is: "+alias.aliasName());
        }

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListAliases in AWS SDK for Java 2.x API Reference](#).

## List grants for a key

The following code example shows how to list grants for a KMS key.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void displayGrantIds(KmsClient kmsClient, String keyId) {

    try {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
```

```
.limit(15)
.build();

ListGrantsResponse response = kmsClient.listGrants(grantsRequest);
List<GrantListEntry> grants = response.grants();
for ( GrantListEntry grant: grants) {
    System.out.println("The grant Id is : "+grant.grantId());
}

} catch (KmsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [ListGrants](#) in *AWS SDK for Java 2.x API Reference*.

## List keys

The following code example shows how to list KMS keys.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllKeys(KmsClient kmsClient) {

    try {
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        ListKeysResponse keysResponse = kmsClient.listKeys(listKeysRequest);
        List<KeyListEntry> keyListEntries = keysResponse.keys();
        for (KeyListEntry key : keyListEntries) {
            System.out.println("The key ARN is: " + key.keyArn());
            System.out.println("The key Id is: " + key.keyId());
        }

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListKeys](#) in *AWS SDK for Java 2.x API Reference*.

## Kinesis examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Kinesis.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Topics

- [Actions \(p. 3309\)](#)

## Actions

### Create a stream

The following code example shows how to create a Kinesis stream.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createStream(KinesisClient kinesisClient, String streamName) {  
  
    try {  
        CreateStreamRequest streamReq = CreateStreamRequest.builder()  
            .streamName(streamName)  
            .shardCount(1)  
            .build();  
  
        kinesisClient.createStream(streamReq);  
  
    } catch (KinesisException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [CreateStream](#) in *AWS SDK for Java 2.x API Reference*.

### Delete a stream

The following code example shows how to delete a Kinesis stream.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteStream(KinesisClient kinesisClient, String streamName) {  
  
    try {  
        DeleteStreamRequest delStream = DeleteStreamRequest.builder()  
            .streamName(streamName)  
            .build();  
  
        kinesisClient.deleteStream(delStream);  
  
    } catch (KinesisException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteStream](#) in *AWS SDK for Java 2.x API Reference*.

## Get data in batches from a stream

The following code example shows how to get data in batches from a Kinesis stream.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getStockTrades(KinesisClient kinesisClient, String streamName) {  
  
    String shardIterator;  
    String lastShardId = null;  
  
    // Retrieve the Shards from a Stream  
    DescribeStreamRequest describeStreamRequest = DescribeStreamRequest.builder()  
        .streamName(streamName)  
        .build();  
  
    List<Shard> shards = new ArrayList<>();  
    DescribeStreamResponse streamRes;  
    do {  
        streamRes = kinesisClient.describeStream(describeStreamRequest);  
        shards.addAll(streamRes.streamDescription().shards());  
  
        if (shards.size() > 0) {  
            lastShardId = shards.get(shards.size() - 1).shardId();  
        }  
    } while (streamRes.streamDescription().hasMoreShards());  
  
    GetShardIteratorRequest itReq = GetShardIteratorRequest.builder()  
        .streamName(streamName)  
        .shardIteratorType("TRIM_HORIZON")  
        .shardId(lastShardId)  
        .build();  
  
    GetShardIteratorResponse shardIteratorResult =  
    kinesisClient.getShardIterator(itReq);  
    shardIterator = shardIteratorResult.shardIterator();  
  
    // Continuously read data records from shard.  
    List<Record> records;  
  
    // Create new GetRecordsRequest with existing shardIterator.  
    // Set maximum records to return to 1000.  
    GetRecordsRequest recordsRequest = GetRecordsRequest.builder()  
        .shardIterator(shardIterator)  
        .limit(1000)  
        .build();  
  
    GetRecordsResponse result = kinesisClient.getRecords(recordsRequest);  
  
    // Put result into record list. Result may be empty.  
    records = result.records();  
  
    // Print records  
    for (Record record : records) {  
        SdkBytes byteBuffer = record.data();  
    }  
}
```

```
        System.out.printf("Seq No: %s - %s%n", record.sequenceNumber(), new
String(byteBuffer.asarray()));
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [GetRecords](#)
  - [GetShardIterator](#)

## Put data into a stream

The following code example shows how to put data into a Kinesis stream.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void setStockData( KinesisClient kinesisClient, String streamName) {

    try {
        // Repeatedly send stock trades with a 100 milliseconds wait in between.
        StockTradeGenerator stockTradeGenerator = new StockTradeGenerator();

        // Put in 50 Records for this example.
        int index = 50;
        for (int x=0; x<index; x++){
            StockTrade trade = stockTradeGenerator.getRandomTrade();
            sendStockTrade(trade, kinesisClient, streamName);
            Thread.sleep(100);
        }

    } catch (KinesisException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done");
}

private static void sendStockTrade(StockTrade trade, KinesisClient kinesisClient,
                                 String streamName) {
    byte[] bytes = trade.toJsonAsBytes();

    // The bytes could be null if there is an issue with the JSON serialization by
    // the Jackson JSON library.
    if (bytes == null) {
        System.out.println("Could not get JSON bytes for stock trade");
        return;
    }

    System.out.println("Putting trade: " + trade);
    PutRecordRequest request = PutRecordRequest.builder()
        .partitionKey(trade.getTickerSymbol()) // We use the ticker symbol as the
                                                // partition key, explained in the Supplemental Information section below.
        .streamName(streamName)
        .data(SdkBytes.fromByteArray(bytes))
        .build();

    try {
        kinesisClient.putRecord(request);
```

```
        } catch (KinesisException e) {
            e.getMessage();
        }
    }

    private static void validateStream(KinesisClient kinesisClient, String streamName)
{
    try {
        DescribeStreamRequest describeStreamRequest =
DescribeStreamRequest.builder()
        .streamName(streamName)
        .build();

        DescribeStreamResponse describeStreamResponse =
kinesisClient.describeStream(describeStreamRequest);

        if(!
describeStreamResponse.streamDescription().streamStatus().toString().equals("ACTIVE"))
{
            System.err.println("Stream " + streamName + " is not active. Please
wait a few moments and try again.");
            System.exit(1);
}

    }catch (KinesisException e) {
        System.err.println("Error found while describing the stream " +
streamName);
        System.err.println(e);
        System.exit(1);
}
}
```

- For API details, see [PutRecord](#) in *AWS SDK for Java 2.x API Reference*.

## Lambda examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS Lambda.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3312\)](#)
- [Scenarios \(p. 3314\)](#)

## Actions

### Create a function

The following code example shows how to create a Lambda function.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createLambdaFunction(LambdaClient awsLambda,
                                         String functionName,
                                         String filePath,
                                         String role,
                                         String handler) {

    try {
        LambdaWaiter waiter = awsLambda.waiter();
        InputStream is = new FileInputStream(filePath);
        SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

        FunctionCode code = FunctionCode.builder()
            .zipFile(fileToUpload)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .functionName(functionName)
            .description("Created by the Lambda Java API")
            .code(code)
            .handler(handler)
            .runtime(Runtime.JAVA8)
            .role(role)
            .build();

        // Create a Lambda function using a waiter.
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();
        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The function ARN is " +
functionResponse.functionArn());

    } catch(LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateFunction](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a function

The following code example shows how to delete a Lambda function.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();
```

```
        awsLambda.deleteFunction(request);
        System.out.println("The "+functionName +" function was deleted");

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteFunction](#) in *AWS SDK for Java 2.x API Reference*.

## Invoke a function

The following code example shows how to invoke a Lambda function.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res = null ;
    try {
        // Need a SdkBytes instance for the payload.
        JSONObject jsonObj = new JSONObject();
        jsonObj.put("inputValue", "2000");
        String json = jsonObj.toString();
        SdkBytes payload = SdkBytes.fromUtf8String(json) ;

        // Setup an InvokeRequest.
        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String() ;
        System.out.println(value);

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [Invoke](#) in *AWS SDK for Java 2.x API Reference*.

## Scenarios

### Get started with functions

The following code example shows how to:

- Create an AWS Identity and Access Management (IAM) role that grants Lambda permission to write to logs.

- Create a Lambda function and upload handler code.
- Invoke the function with a single parameter and get results.
- Update the function code and configure its Lambda environment with an environment variable.
- Invoke the function with new parameters and get results. Display the execution log that's returned from the invocation.
- List the functions for your account.
- Delete the IAM role and the Lambda function.

For more information, see [Create a Lambda function with the console](#).

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/*
 * Lambda function names appear as:
 *
 * arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
 *
 * To find this value, look at the function in the AWS Management Console.
 *
 * Before running this Java code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example performs the following tasks:
 *
 * 1. Creates an AWS Lambda function.
 * 2. Gets a specific AWS Lambda function.
 * 3. Lists all Lambda functions.
 * 4. Invokes a Lambda function.
 * 5. Updates the Lambda function code and invokes it again.
 * 6. Updates a Lambda function's configuration value.
 * 7. Deletes a Lambda function.
 */

public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws InterruptedException {

        final String usage = "\n" +
            "Usage:\n" +
            "      <functionName> <filePath> <role> <handler> <bucketName> <key> \n\n" +
            "Where:\n" +
            "      functionName - The name of the Lambda function. \n" +
            "      filePath - The path to the .zip or .jar where the code is located.
\n" +
            "      role - The AWS Identity and Access Management (IAM) service role that
has Lambda permissions. \n" +
            "      handler - The fully qualified method name (for example,
example.Handler::handleRequest). \n" +
            "      bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name
that contains the .zip or .jar used to update the Lambda function's code. \n" +
            "      key - The Amazon S3 key name that represents the .zip or .jar (for
example, LambdaHello-1.0-SNAPSHOT.jar)." ;
    }
}
```

```
if (args.length != 6) {
    System.out.println(usage);
    System.exit(1);
}

String functionName = args[0];
String filePath = args[1];
String role = args[2];
String handler = args[3];
String bucketName = args[4];
String key = args[5];

Region region = Region.US_WEST_2;
LambdaClient awsLambda = LambdaClient.builder()
    .region(region)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the AWS Lambda example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an AWS Lambda function.");
String funArn = createLambdaFunction(awsLambda, functionName, filePath, role,
handler);
System.out.println("The AWS Lambda ARN is "+funArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Get the "+functionName + " AWS Lambda function.");
getFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. List all AWS Lambda functions.");
listFunctions(awsLambda);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Invoke the Lambda function.");
System.out.println("*** Sleep for 1 min to get Lambda function ready.");
Thread.sleep(60000);
invokeFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Update the Lambda function code and invoke it again.");
updateFunctionCode(awsLambda, functionName, bucketName, key);
System.out.println("*** Sleep for 1 min to get Lambda function ready.");
Thread.sleep(60000);
invokeFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Update a Lambda function's configuration value.");
updateFunctionConfiguration(awsLambda, functionName, handler);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the AWS Lambda function.");
LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
        System.out.println("The AWS Lambda scenario completed successfully");
        System.out.println(DASHES);
        awsLambda.close();
    }

    public static String createLambdaFunction(LambdaClient awsLambda,
                                              String functionName,
                                              String filePath,
                                              String role,
                                              String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

            FunctionCode code = FunctionCode.builder()
                .zipFile(fileToUpload)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA8)
                .role(role)
                .build();

            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch(LambdaException | FileNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static void getFunction(LambdaClient awsLambda, String functionName) {
        try {
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();

            GetFunctionResponse response = awsLambda.getFunction(functionRequest);
            System.out.println("The runtime of this Lambda function is "
+response.configuration().runtime());

        } catch(LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
```

```
        List<FunctionConfiguration> list = functionResult.functions();
        for (FunctionConfiguration config: list) {
            System.out.println("The function name is "+config.functionName());
        }

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res;
    try {
        // Need a SdkBytes instance for the payload.
        JSONObject jsonObj = new JSONObject();
        jsonObj.put("inputValue", "2000");
        String json = jsonObj.toString();
        SdkBytes payload = SdkBytes.fromUtf8String(json) ;

        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
            .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String() ;
        System.out.println(value);

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateFunctionCode(LambdaClient awsLambda, String functionName,
String bucketName, String key) {
    try {
        LambdaWaiter waiter = awsLambda.waiter();
        UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
            .functionName(functionName)
            .publish(true)
            .s3Bucket(bucketName)
            .s3Key(key)
            .build();

        UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest) ;
        GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
            .functionName(functionName)
            .build();

        WaiterResponse<GetFunctionConfigurationResponse> waiterResponse =
waiter.waitUntilFunctionUpdated(getFunctionConfigRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The last modified value is " +response.lastModified());

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler ) {
    try {
        UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
            .functionName(functionName)
            .handler(handler)
            .runtime(Runtime.JAVA11 )
            .build();

        awsLambda.updateFunctionConfiguration(configurationRequest);

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName ) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("The "+functionName +" function was deleted");

    } catch(LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.

- [CreateFunction](#)
- [DeleteFunction](#)
- [GetFunction](#)
- [Invoke](#)
- [ListFunctions](#)
- [UpdateFunctionCode](#)
- [UpdateFunctionConfiguration](#)

## Amazon Personalize examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Personalize.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3320\)](#)

## Actions

### Create a batch interface job

The following code example shows how to create a Amazon Personalize batch interface job.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                                                       String solutionVersionArn,
                                                       String jobName,
                                                       String
s3InputDataSourcePath,
                                                       String
s3DataDestinationPath,
                                                       String
                                                       String roleArn,
                                                       String explorationWeight,
                                                       String
explorationItemAgeCutOff) {

    long waitInMilliseconds = 60 * 1000;
    String status;
    String batchInferenceJobArn;

    try {

        // Set up data input and output parameters.
        S3DataConfig inputSource = S3DataConfig.builder()
            .path(s3InputDataSourcePath)
            .build();

        S3DataConfig outputDestination = S3DataConfig.builder()
            .path(s3DataDestinationPath)
            .build();

        BatchInferenceJobInput jobInput = BatchInferenceJobInput.builder()
            .s3DataSource(inputSource)
            .build();

        BatchInferenceJobOutput jobOutputLocation =
BatchInferenceJobOutput.builder()
            .s3DataDestination(outputDestination)
            .build();

        // Optional code to build the User-Personalization specific item
exploration config.
        HashMap<String, String> explorationConfig = new HashMap<>();

        explorationConfig.put("explorationWeight", explorationWeight);
        explorationConfig.put("explorationItemAgeCutOff",
explorationItemAgeCutOff);

        BatchInferenceJobConfig jobConfig = BatchInferenceJobConfig.builder()
            .itemExplorationConfig(explorationConfig)
            .build();

        // End optional User-Personalization recipe specific code.
```

```
CreateBatchInferenceJobRequest createBatchInferenceJobRequest =  
CreateBatchInferenceJobRequest.builder()  
.solutionVersionArn(solutionVersionArn)  
.jobInput(jobInput)  
.jobOutput(jobOutputLocation)  
.jobName(jobName)  
.roleArn(roleArn)  
.batchInferenceJobConfig(jobConfig) // Optional  
.build();  
  
batchInferenceJobArn =  
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)  
.batchInferenceJobArn();  
  
DescribeBatchInferenceJobRequest describeBatchInferenceJobRequest =  
DescribeBatchInferenceJobRequest.builder()  
.batchInferenceJobArn(batchInferenceJobArn)  
.build();  
  
long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;  
while (Instant.now().getEpochSecond() < maxTime) {  
  
    BatchInferenceJob batchInferenceJob = personalizeClient  
.describeBatchInferenceJob(describeBatchInferenceJobRequest)  
.batchInferenceJob();  
  
    status = batchInferenceJob.status();  
    System.out.println("Batch inference job status: " + status);  
  
    if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {  
        break;  
    }  
    try {  
        Thread.sleep(waitInMilliseconds);  
    } catch (InterruptedException e) {  
        System.out.println(e.getMessage());  
    }  
}  
return batchInferenceJobArn;  
  
} catch (PersonalizeException e) {  
    System.out.println(e.awsErrorDetails().errorMessage());  
}  
return "";  
}
```

- For API details, see [CreateBatchInferenceJob](#) in *AWS SDK for Java 2.x API Reference*.

## Create a campaign

The following code example shows how to create a Amazon Personalize campaign.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createPersonalCompaign(PersonalizeClient personalizeClient,  
String solutionVersionArn, String name) {  
  
    try {
```

```
CreateCampaignRequest createCampaignRequest =  
CreateCampaignRequest.builder()  
    .minProvisionedTPS(1)  
    .solutionVersionArn(solutionVersionArn)  
    .name(name)  
    .build();  
  
CreateCampaignResponse campaignResponse =  
personalizeClient.createCampaign(createCampaignRequest);  
System.out.println("The campaign ARN is "+campaignResponse.campaignArn());  
  
} catch (PersonalizeException e) {  
    System.err.println(e.awsErrorDetails().errorMessage());  
    System.exit(1);  
}  
}
```

- For API details, see [CreateCampaign in AWS SDK for Java 2.x API Reference](#).

## Create a dataset

The following code example shows how to create a Amazon Personalize dataset.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createDataset(PersonalizeClient personalizeClient,  
                                    String datasetName,  
                                    String datasetGroupArn,  
                                    String datasetType,  
                                    String schemaArn) {  
    try {  
        CreateDatasetRequest request = CreateDatasetRequest.builder()  
            .name(datasetName)  
            .datasetGroupArn(datasetGroupArn)  
            .datasetType(datasetType)  
            .schemaArn(schemaArn)  
            .build();  
  
        String datasetArn = personalizeClient.createDataset(request)  
            .datasetArn();  
        System.out.println("Dataset " + datasetName + " created.");  
        return datasetArn;  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [CreateDataset in AWS SDK for Java 2.x API Reference](#).

## Create a dataset export job

The following code example shows how to create a Amazon Personalize dataset export job.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createDatasetExportJob(PersonalizeClient personalizeClient,
                                             String jobName,
                                             String datasetArn,
                                             IngestionMode ingestionMode,
                                             String roleArn,
                                             String s3BucketPath,
                                             String kmsKeyArn) {

    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String status = null;

    try {

        S3DataConfig exportS3DataConfig =
S3DataConfig.builder().path(s3BucketPath).kmsKeyArn(kmsKeyArn).build();
        DatasetExportJobOutput jobOutput =
DatasetExportJobOutput.builder().s3DataDestination(exportS3DataConfig).build();

        CreateDatasetExportJobRequest createRequest =
CreateDatasetExportJobRequest.builder()
            .jobName(jobName)
            .datasetArn(datasetArn)
            .ingestionMode(ingestionMode)
            .jobOutput(jobOutput)
            .roleArn(roleArn)
            .build();

        String datasetExportJobArn =
personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

        DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
DescribeDatasetExportJobRequest.builder()
            .datasetExportJobArn(datasetExportJobArn)
            .build();

        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        while (Instant.now().getEpochSecond() < maxTime) {

            DatasetExportJob datasetExportJob =
personalizeClient.describeDatasetExportJob(describeDatasetExportJobRequest)
            .datasetExportJob();

            status = datasetExportJob.status();
            System.out.println("Export job status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                return status;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        return "";
    }
```

- For API details, see [CreateDatasetExportJob](#) in *AWS SDK for Java 2.x API Reference*.

## Create a dataset group

The following code example shows how to create a Amazon Personalize dataset group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createDatasetGroup(PersonalizeClient personalizeClient, String
datasetGroupName) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
            .name(datasetGroupName)
            .build();
        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

Create a domain dataset group.

```
public static String createDomainDatasetGroup(PersonalizeClient personalizeClient,
                                              String datasetGroupName,
                                              String domain) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
            .name(datasetGroupName)
            .domain(domain)
            .build();
        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

- For API details, see [CreateDatasetGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Create a dataset import job

The following code example shows how to create a Amazon Personalize dataset import job.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,
                                                       String jobName,
                                                       String datasetArn,
                                                       String s3BucketPath,
                                                       String roleArn) {

    long waitInMilliseconds = 60 * 1000;
    String status;
    String datasetImportJobArn;

    try {
        DataSource importDataSource = DataSource.builder()
            .dataLocation(s3BucketPath)
            .build();

        CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
            .datasetArn(datasetArn)
            .dataSource(importDataSource)
            .jobName(jobName)
            .roleArn(roleArn)
            .build();

        datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
            .datasetImportJobArn();
        DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
            .datasetImportJobArn(datasetImportJobArn)
            .build();

        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        while (Instant.now().getEpochSecond() < maxTime) {

            DatasetImportJob datasetImportJob = personalizeClient
                .describeDatasetImportJob(describeDatasetImportJobRequest)
                .datasetImportJob();

            status = datasetImportJob.status();
            System.out.println("Dataset import job status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        return datasetImportJobArn;
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

```
}
```

- For API details, see [CreateDatasetImportJob](#) in *AWS SDK for Java 2.x API Reference*.

## Create a domain schema

The following code example shows how to create a Amazon Personalize domain schema.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createDomainSchema(PersonalizeClient personalizeClient, String schemaName, String domain, String filePath) {  
  
    String schema = null;  
    try {  
        schema = new String(Files.readAllBytes(Paths.get(filePath)));  
    } catch (IOException e) {  
        System.out.println(e.getMessage());  
    }  
  
    try {  
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()  
            .name(schemaName)  
            .domain(domain)  
            .schema(schema)  
            .build();  
  
        String schemaArn =  
personalizeClient.createSchema(createSchemaRequest).schemaArn();  
  
        System.out.println("Schema arn: " + schemaArn);  
  
        return schemaArn;  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [CreateSchema](#) in *AWS SDK for Java 2.x API Reference*.

## Create a filter

The following code example shows how to create a Amazon Personalize filter.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createFilter(PersonalizeClient personalizeClient,
```

```
        String filterName,
        String datasetGroupArn,
        String filterExpression) {
    try {
        CreateFilterRequest request = CreateFilterRequest.builder()
            .name(filterName)
            .datasetGroupArn(datasetGroupArn)
            .filterExpression(filterExpression)
            .build();

        return personalizeClient.createFilter(request).filterArn();
    }
    catch(PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateFilter](#) in *AWS SDK for Java 2.x API Reference*.

## Create a recommender

The following code example shows how to create a Amazon Personalize recommender.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createRecommender(PersonalizeClient personalizeClient,
                                         String name,
                                         String datasetGroupArn,
                                         String recipeArn) {

    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String recommenderStatus = "";

    try {
        CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
            .datasetGroupArn(datasetGroupArn)
            .name(name)
            .recipeArn(recipeArn)
            .build();

        CreateRecommenderResponse recommenderResponse =
personalizeClient.createRecommender(createRecommenderRequest);
        String recommenderArn = recommenderResponse.recommenderArn();
        System.out.println("The recommender ARN is " + recommenderArn);

        DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
            .recommenderArn(recommenderArn)
            .build();

        maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        while (Instant.now().getEpochSecond() < maxTime) {
```

```
        recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender().status();
        System.out.println("Recommender status: " + recommenderStatus);

        if (recommenderStatus.equals("ACTIVE") ||
recommenderStatus.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return recommenderArn;

} catch(PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [CreateRecommender](#) in *AWS SDK for Java 2.x API Reference*.

## Create a schema

The following code example shows how to create a Amazon Personalize schema.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

    String schema = null;
    try {
        schema = new String(Files.readAllBytes(Paths.get(filePath)));
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }

    try {
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
            .name(schemaName)
            .schema(schema)
            .build();

        String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

        System.out.println("Schema arn: " + schemaArn);

        return schemaArn;
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return "";
    }
```

- For API details, see [CreateSchema](#) in *AWS SDK for Java 2.x API Reference*.

## Create a solution

The following code example shows how to create a Amazon Personalize solution.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createPersonalizeSolution(PersonalizeClient personalizeClient,
                                                String datasetGroupArn,
                                                String solutionName,
                                                String recipeArn) {

    try {
        CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
            .name(solutionName)
            .datasetGroupArn(datasetGroupArn)
            .recipeArn(recipeArn)
            .build();

        CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
        return solutionResponse.solutionArn();

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateSolution](#) in *AWS SDK for Java 2.x API Reference*.

## Create a solution version

The following code example shows how to create a Amazon Personalize solution.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String solutionStatus = "";
    String solutionVersionStatus = "";
    String solutionVersionArn = "";
```

```
try {
    DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
    .solutionArn(solutionArn)
    .build();

    maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    // Wait until solution is active.
    while (Instant.now().getEpochSecond() < maxTime) {

        solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
        System.out.println("Solution status: " + solutionStatus);

        if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }

    if (solutionStatus.equals("ACTIVE")) {

        CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
    .solutionArn(solutionArn)
    .build();

        CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient.createSolutionVersion(createSolutionVersionRequest);
        solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

        System.out.println("Solution version ARN: " + solutionVersionArn);

        DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
    .solutionVersionArn(solutionVersionArn)
    .build();

        while (Instant.now().getEpochSecond() < maxTime) {

            solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest).solutionVersion().status();
            System.out.println("Solution version status: " +
solutionVersionStatus);

            if (solutionVersionStatus.equals("ACTIVE") || solutionVersionStatus.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        return solutionVersionArn;
    }
} catch(PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}
```

```
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateSolutionVersion](#) in *AWS SDK for Java 2.x API Reference*.

## Create an event tracker

The following code example shows how to create a Amazon Personalize event tracker.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createEventTracker(PersonalizeClient personalizeClient, String
eventTrackerName, String datasetGroupArn) {

    String eventTrackerId = "";
    String eventTrackerArn;
    long maxTime = 3 * 60 * 60; // 3 hours
    long waitInMilliseconds = 20 * 1000; // 20 seconds
    String status;

    try {

        CreateEventTrackerRequest createEventTrackerRequest =
CreateEventTrackerRequest.builder()
            .name(eventTrackerName)
            .datasetGroupArn(datasetGroupArn)
            .build();

        CreateEventTrackerResponse createEventTrackerResponse =
personalizeClient.createEventTracker(createEventTrackerRequest);

        eventTrackerArn = createEventTrackerResponse.eventTrackerArn();
        eventTrackerId = createEventTrackerResponse.trackingId();
        System.out.println("Event tracker ARN: " + eventTrackerArn);
        System.out.println("Event tracker ID: " + eventTrackerId);

        maxTime = Instant.now().getEpochSecond() + maxTime;

        DescribeEventTrackerRequest describeRequest =
DescribeEventTrackerRequest.builder()
            .eventTrackerArn(eventTrackerArn)
            .build();

        while (Instant.now().getEpochSecond() < maxTime) {

            status =
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();
            System.out.println("EventTracker status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
```

```
        System.out.println(e.getMessage());
    }
    return eventTrackerId;
}
catch (PersonalizeException e){
    System.out.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return eventTrackerId;
}
```

- For API details, see [CreateEventTracker](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a campaign

The following code example shows how to delete a campaign in Amazon Personalize.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn ) {

    try {
        DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        personalizeClient.deleteCampaign(campaignRequest);

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteCampaign](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a solution

The following code example shows how to delete a solution in Amazon Personalize.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteGivenSolution(PersonalizeClient personalizeClient, String
solutionArn ) {

    try {
        DeleteSolutionRequest solutionRequest = DeleteSolutionRequest.builder()
            .solutionArn(solutionArn)
```

```
        .build();

    personalizeClient.deleteSolution(solutionRequest);
    System.out.println("Done");

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [DeleteSolution](#) in *AWS SDK for Java 2.x API Reference*.

## Delete an event tracker

The following code example shows how to delete an event tracker in Amazon Personalize.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteEventTracker(PersonalizeClient personalizeClient, String
eventTrackerArn) {
    try {
        DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
            .eventTrackerArn(eventTrackerArn)
            .build();

        int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCode();

        System.out.println("Status code:" + status);

    }
    catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteEventTracker](#) in *AWS SDK for Java 2.x API Reference*.

## Describe a campaign

The following code example shows how to describe a campaign in Amazon Personalize.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {
```

```
try {
    DescribeCampaignRequest campaignRequest = DescribeCampaignRequest.builder()
        .campaignArn(campaignArn)
        .build();

    DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
    Campaign myCampaign = campaignResponse.campaign();
    System.out.println("The Campaign name is "+myCampaign.name());
    System.out.println("The Campaign status is "+myCampaign.status());

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [DescribeCampaign](#) in *AWS SDK for Java 2.x API Reference*.

## Describe a recipe

The following code example shows how to describe a recipe in Amazon Personalize.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeSpecificRecipe(PersonalizeClient personalizeClient,
String recipeArn) {

    try{
        DescribeRecipeRequest recipeRequest = DescribeRecipeRequest.builder()
            .recipeArn(recipeArn)
            .build();

        DescribeRecipeResponse recipeResponse =
personalizeClient.describeRecipe(recipeRequest);
        System.out.println("The recipe name is "+recipeResponse.recipe().name());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeRecipe](#) in *AWS SDK for Java 2.x API Reference*.

## Describe a solution

The following code example shows how to describe a solution in Amazon Personalize.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeSpecificSolution(PersonalizeClient personalizeClient,  
String solutionArn) {  
  
    try {  
        DescribeSolutionRequest solutionRequest = DescribeSolutionRequest.builder()  
            .solutionArn(solutionArn)  
            .build();  
  
        DescribeSolutionResponse response =  
personalizeClient.describeSolution(solutionRequest);  
        System.out.println("The Solution name is "+response.solution().name());  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DescribeSolution](#) in *AWS SDK for Java 2.x API Reference*.

## List campaigns

The following code example shows how to list campaigns in Amazon Personalize.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllCampaigns(PersonalizeClient personalizeClient, String  
solutionArn) {  
  
    try{  
        ListCampaignsRequest campaignsRequest = ListCampaignsRequest.builder()  
            .maxResults(10)  
            .solutionArn(solutionArn)  
            .build();  
  
        ListCampaignsResponse response =  
personalizeClient.listCampaigns(campaignsRequest);  
        List<CampaignSummary> campaigns = response.campaigns();  
        for (CampaignSummary campaign: campaigns) {  
            System.out.println("Campaign name is : "+campaign.name());  
            System.out.println("Campaign ARN is : "+campaign.campaignArn());  
        }  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListCampaigns](#) in *AWS SDK for Java 2.x API Reference*.

## List dataset groups

The following code example shows how to list dataset groups in Amazon Personalize.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listDSGroups( PersonalizeClient personalizeClient ) {

    try {
        ListDatasetGroupsRequest groupsRequest = ListDatasetGroupsRequest.builder()
            .maxResults(15)
            .build();

        ListDatasetGroupsResponse groupsResponse =
personalizeClient.listDatasetGroups(groupsRequest);
        List<DatasetGroupSummary> groups = groupsResponse.datasetGroups();
        for (DatasetGroupSummary group: groups) {
            System.out.println("The DataSet name is : "+group.name());
            System.out.println("The DataSet ARN is : "+group.datasetGroupArn());
        }

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListDatasetGroups](#) in *AWS SDK for Java 2.x API Reference*.

## List recipes

The following code example shows how to list recipes in Amazon Personalize.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllRecipes(PersonalizeClient personalizeClient) {

    try {
        ListRecipesRequest recipesRequest = ListRecipesRequest.builder()
            .maxResults(15)
            .build();

        ListRecipesResponse response =
personalizeClient.listRecipes(recipesRequest);
        List<RecipeSummary> recipes = response.recipes();
        for (RecipeSummary recipe: recipes) {
            System.out.println("The recipe ARN is: "+recipe.recipeArn());
            System.out.println("The recipe name is: "+recipe.name());
        }

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListRecipes](#) in *AWS SDK for Java 2.x API Reference*.

## List solutions

The following code example shows how to list solutions in Amazon Personalize.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllSolutions(PersonalizeClient personalizeClient, String datasetGroupArn) {  
  
    try {  
        ListSolutionsRequest solutionsRequest = ListSolutionsRequest.builder()  
            .maxResults(10)  
            .datasetGroupArn(datasetGroupArn)  
            .build();  
  
        ListSolutionsResponse response =  
personalizeClient.listSolutions(solutionsRequest);  
        List<SolutionSummary> solutions = response.solutions();  
        for (SolutionSummary solution: solutions) {  
            System.out.println("The solution ARN is: "+solution.solutionArn());  
            System.out.println("The solution name is: "+solution.name());  
        }  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListSolutions](#) in *AWS SDK for Java 2.x API Reference*.

## Update a campaign

The following code example shows how to update a campaign Amazon Personalize.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String updateCampaign(PersonalizeClient personalizeClient,  
                                    String campaignArn,  
                                    String solutionVersionArn,  
                                    Integer minProvisionedTPS) {  
  
    try {  
        // build the updateCampaignRequest  
        UpdateCampaignRequest updateCampaignRequest =  
UpdateCampaignRequest.builder()  
            .campaignArn(campaignArn)  
            .solutionVersionArn(solutionVersionArn)  
            .minProvisionedTPS(minProvisionedTPS)
```

```
        .build();

    // update the campaign
    personalizeClient.updateCampaign(updateCampaignRequest);

    DescribeCampaignRequest campaignRequest = DescribeCampaignRequest.builder()
        .campaignArn(campaignArn)
        .build();

    DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
    Campaign updatedCampaign = campaignResponse.campaign();

    System.out.println("The Campaign status is " + updatedCampaign.status());
    return updatedCampaign.status();

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

- For API details, see [UpdateCampaign](#) in *AWS SDK for Java 2.x API Reference*.

## Amazon Personalize Events examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Personalize Events.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3338\)](#)

## Actions

### Import real-time interaction event data

The following code example shows how to import real-time interaction event data into Amazon Personalize Events.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static int putItems(PersonalizeEventsClient personalizeEventsClient,
                           String datasetArn,
                           String item1Id,
                           String item1PropertyName,
                           String item1PropertyValue,
                           String item2Id,
                           String item2PropertyName,
                           String item2PropertyValue) {
```

```
int responseCode = 0;
ArrayList<Item> items = new ArrayList<>();

try {
    Item item1 = Item.builder()
        .itemId(item1Id)
        .properties(String.format("{\"%1$s\": \"%2$s\"}",
            item1PropertyName, item1PropertyValue))
        .build();

    items.add(item1);

    Item item2 = Item.builder()
        .itemId(item2Id)
        .properties(String.format("{\"%1$s\": \"%2$s\"}",
            item2PropertyName, item2PropertyValue))
        .build();

    items.add(item2);

    PutItemsRequest putItemsRequest = PutItemsRequest.builder()
        .datasetArn(datasetArn)
        .items(items)
        .build();

    responseCode =
personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
    System.out.println("Response code: " + responseCode);
    return responseCode;
} catch (PersonalizeEventsException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return responseCode;
}
```

- For API details, see [PutEvents](#) in *AWS SDK for Java 2.x API Reference*.

## Incrementally import a user

The following code example shows how to incrementally import a user into Amazon Personalize Events Events.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static int putUsers(PersonalizeEventsClient personalizeEventsClient,
                           String datasetArn,
                           String user1Id,
                           String user1PropertyName,
                           String user1PropertyValue,
                           String user2Id,
                           String user2PropertyName,
                           String user2PropertyValue) {

    int responseCode = 0;
    ArrayList<User> users = new ArrayList<>();
```

```
try {
    User user1 = User.builder()
        .userId(user1Id)
        .properties(String.format("{\"%1$s\": \"%2$s\"}",
            user1PropertyName, user1PropertyValue))
        .build();

    users.add(user1);

    User user2 = User.builder()
        .userId(user2Id)
        .properties(String.format("{\"%1$s\": \"%2$s\"}",
            user2PropertyName, user2PropertyValue))
        .build();

    users.add(user2);

    PutUsersRequest putUsersRequest = PutUsersRequest.builder()
        .datasetArn(datasetArn)
        .users(users)
        .build();

    responseCode =
personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
    System.out.println("Response code: " + responseCode);
    return responseCode;
} catch (PersonalizeEventsException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return responseCode;
}
```

- For API details, see [PutUsers](#) in *AWS SDK for Java 2.x API Reference*.

## Amazon Personalize Runtime examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Personalize Runtime.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3340\)](#)

## Actions

### Get recommendations (custom dataset group)

The following code example shows how to get Amazon Personalize Runtime ranked recommendations.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient  
personalizeRuntimeClient,  
                                                String campaignArn,  
                                                String userId,  
                                                ArrayList<String> items) {  
  
    try {  
        GetPersonalizedRankingRequest rankingRecommendationsRequest =  
GetPersonalizedRankingRequest.builder()  
            .campaignArn(campaignArn)  
            .userId(userId)  
            .inputList(items)  
            .build();  
  
        GetPersonalizedRankingResponse recommendationsResponse =  
personalizeRuntimeClient.getPersonalizedRanking(rankingRecommendationsRequest);  
        List<PredictedItem> rankedItems =  
recommendationsResponse.personalizedRanking();  
        int rank = 1;  
        for (PredictedItem item : rankedItems) {  
            System.out.println("Item ranked at position " + rank + " details");  
            System.out.println("Item Id is : " + item.itemId());  
            System.out.println("Item score is : " + item.score());  
            System.out.println("-----");  
            rank++;  
        }  
        return rankedItems;  
    } catch (PersonalizeRuntimeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return null;  
}
```

- For API details, see [GetPersonalizedRanking](#) in *AWS SDK for Java 2.x API Reference*.

## Get recommendations from a recommender (domain dataset group)

The following code example shows how to get Amazon Personalize Runtime Runtime recommendations.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get a list of recommended items.

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,  
String campaignArn, String userId){  
  
    try {  
        GetRecommendationsRequest recommendationsRequest =  
GetRecommendationsRequest.builder()  
            .campaignArn(campaignArn)  
            .numResults(20)  
            .userId(userId)  
            .build();  
    }
```

```

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient.getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();
        for (PredictedItem item: items) {
            System.out.println("Item Id is : "+item.itemId());
            System.out.println("Item score is : "+item.score());
        }
    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

Get a list of recommended items from a recommender created in a domain dataset group.

```

public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String recommenderArn, String userId){

    try {
        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
        .recommenderArn(recommenderArn)
        .numResults(20)
        .userId(userId)
        .build();

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient.getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();

        for (PredictedItem item: items) {
            System.out.println("Item Id is : "+item.itemId());
            System.out.println("Item score is : "+item.score());
        }
    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

Use a filter when requesting recommendations.

```

public static void getFilteredRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
                                         String campaignArn,
                                         String userId,
                                         String filterArn,
                                         String parameter1Name,
                                         String parameter1Value1,
                                         String parameter1Value2,
                                         String parameter2Name,
                                         String parameter2Value){

    try {

        Map<String, String> filterValues = new HashMap<>();

        filterValues.put(parameter1Name, String.format("\"%1$s\", \"%2$s\"",
parameter1Value1, parameter1Value2));
        filterValues.put(parameter2Name, String.format("\"%1$s\"",


```

```
        parameter2Value));  
  
        GetRecommendationsRequest recommendationsRequest =  
GetRecommendationsRequest.builder()  
            .campaignArn(campaignArn)  
            .numResults(20)  
            .userId(userId)  
            .filterArn(filterArn)  
            .filterValues(filterValues)  
            .build();  
  
        GetRecommendationsResponse recommendationsResponse =  
personalizeRuntimeClient.getRecommendations(recommendationsRequest);  
        List<PredictedItem> items = recommendationsResponse.itemList();  
  
        for (PredictedItem item: items) {  
            System.out.println("Item Id is : "+item.itemId());  
            System.out.println("Item score is : "+item.score());  
        }  
    } catch (PersonalizeRuntimeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [GetRecommendations](#) in *AWS SDK for Java 2.x API Reference*.

## Amazon Pinpoint examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Pinpoint.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3343\)](#)

## Actions

### Create a campaign

The following code example shows how to create a campaign.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a campaign.

```
public static void createPinCampaign(PinpointClient pinpoint, String appId, String  
segmentId) {  
    CampaignResponse result = createCampaign(pinpoint, appId, segmentId);  
    System.out.println("Campaign " + result.name() + " created.");  
    System.out.println(result.description());  
}
```

```
public static CampaignResponse createCampaign(PinpointClient client, String appID,
String segmentID) {

    try {
        Schedule schedule = Schedule.builder()
            .startTime("IMMEDIATE")
            .build();

        Message defaultMessage = Message.builder()
            .action(Action.OPEN_APP)
            .body("My message body.")
            .title("My message title.")
            .build();

        MessageConfiguration messageConfiguration = MessageConfiguration.builder()
            .defaultMessage(defaultMessage)
            .build();

        WriteCampaignRequest request = WriteCampaignRequest.builder()
            .description("My description")
            .schedule(schedule)
            .name("MyCampaign")
            .segmentId(segmentID)
            .messageConfiguration(messageConfiguration)
            .build();

        CreateCampaignResponse result =
client.createCampaign(CreateCampaignRequest.builder()
                    .applicationId(appID)
                    .writeCampaignRequest(request).build()
);
        System.out.println("Campaign ID: " + result.campaignResponse().id());
        return result.campaignResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return null;
}
```

- For API details, see [CreateCampaign in AWS SDK for Java 2.x API Reference](#).

## Create a segment

The following code example shows how to create a segment.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static SegmentResponse createSegment(PinpointClient client, String appId) {

    try {
        Map<String, AttributeDimension> segmentAttributes = new HashMap<>();
        segmentAttributes.put("Team", AttributeDimension.builder()
            .attributeType(AttributeType.INCLUSIVE)
```

```
.values("Lakers")
.build();

RecencyDimension recencyDimension = RecencyDimension.builder()
.duration("DAY_30")
.recencyType("ACTIVE")
.build();

SegmentBehaviors segmentBehaviors = SegmentBehaviors.builder()
.recency(recencyDimension)
.build();

SegmentDemographics segmentDemographics = SegmentDemographics
.builder()
.build();

SegmentLocation segmentLocation = SegmentLocation
.builder()
.build();

SegmentDimensions dimensions = SegmentDimensions
.builder()
.attributes(segmentAttributes)
.behavior(segmentBehaviors)
.demographic(segmentDemographics)
.location(segmentLocation)
.build();

WriteSegmentRequest writeSegmentRequest = WriteSegmentRequest.builder()
.name("MySegment")
.dimensions(dimensions)
.build();

CreateSegmentRequest createSegmentRequest = CreateSegmentRequest.builder()
.applicationId(appId)
.writeSegmentRequest(writeSegmentRequest)
.build();

CreateSegmentResponse createSegmentResult =
client.createSegment(createSegmentRequest);
System.out.println("Segment ID: " +
createSegmentResult.segmentResponse().id());
System.out.println("Done");
return createSegmentResult.segmentResponse();

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}
```

- For API details, see [CreateSegment in AWS SDK for Java 2.x API Reference](#).

## Create an application

The following code example shows how to create an application.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createApplication(PinpointClient pinpoint, String appName) {  
  
    try {  
        CreateApplicationRequest appRequest = CreateApplicationRequest.builder()  
            .name(appName)  
            .build();  
  
        CreateAppRequest request = CreateAppRequest.builder()  
            .createApplicationRequest(appRequest)  
            .build();  
  
        CreateAppResponse result = pinpoint.createApp(request);  
        return result.applicationResponse().id();  
  
    } catch (PinpointException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [CreateApp](#) in *AWS SDK for Java 2.x API Reference*.

## Delete an application

The following code example shows how to delete an application.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an application.

```
public static void deletePinApp(PinpointClient pinpoint, String appId ) {  
  
    try {  
        DeleteAppRequest appRequest = DeleteAppRequest.builder()  
            .applicationId(appId)  
            .build();  
  
        DeleteAppResponse result = pinpoint.deleteApp(appRequest);  
        String appName = result.applicationResponse().name();  
        System.out.println("Application " + appName + " has been deleted.");  
  
    } catch (PinpointException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DeleteApp](#) in *AWS SDK for Java 2.x API Reference*.

## Delete an endpoint

The following code example shows how to delete an endpoint.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an endpoint.

```
public static void deletePinEncpoint(PinpointClient pinpoint, String appId, String endpointId) {  
  
    try {  
        DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()  
            .applicationId(appId)  
            .endpointId(endpointId)  
            .build();  
  
        DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);  
        String id = result.endpointResponse().id();  
        System.out.println("The deleted endpoint id " + id);  
  
    } catch (PinpointException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    System.out.println("Done");  
}
```

- For API details, see [DeleteEndpoint](#) in *AWS SDK for Java 2.x API Reference*.

## Export an endpoint

The following code example shows how to export an endpoint.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Export an endpoint.

```
public static void exportAllEndpoints(PinpointClient pinpoint,  
                                     S3Client s3Client,  
                                     String applicationId,  
                                     String s3BucketName,  
                                     String path,  
                                     String iamExportRoleArn) {  
  
    try {  
        List<String> objectKeys = exportEndpointsToS3(pinpoint, s3Client,  
            s3BucketName, iamExportRoleArn, applicationId);  
        List<String> endpointFileKeys = objectKeys.stream().filter(o ->  
            o.endsWith(".gz")).collect(Collectors.toList());  
        downloadFromS3(s3Client, path, s3BucketName, endpointFileKeys);  
  
    } catch ( PinpointException e ) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

```

    public static List<String> exportEndpointsToS3(PinpointClient pinpoint, S3Client
s3Client, String s3BucketName, String iamExportRoleArn, String applicationId) {

        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd-
HH_mm:ss.SSS_z");
        String endpointsKeyPrefix = "exports/" + applicationId + "_" +
dateFormat.format(new Date());
        String s3UrlPrefix = "s3://" + s3BucketName + "/" + endpointsKeyPrefix + "/";
        List<String> objectKeys = new ArrayList<>();
        String key;

        try {
            // Defines the export job that Amazon Pinpoint runs.
            ExportJobRequest jobRequest = ExportJobRequest.builder()
                .roleArn(iamExportRoleArn)
                .s3UrlPrefix(s3UrlPrefix)
                .build();

            CreateExportJobRequest exportJobRequest = CreateExportJobRequest.builder()
                .applicationId(applicationId)
                .exportJobRequest(jobRequest)
                .build();

            System.out.format("Exporting endpoints from Amazon Pinpoint application %s
to Amazon S3 " +
                    "bucket %s . .\n", applicationId, s3BucketName);

            CreateExportJobResponse exportResult =
pinpoint.createExportJob(exportJobRequest);
            String jobId = exportResult.exportJobResponse().id();
            System.out.println(jobId);
            printExportJobStatus(pinpoint, applicationId, jobId);

            ListObjectsV2Request v2Request = ListObjectsV2Request.builder()
                .bucket(s3BucketName)
                .prefix(endpointsKeyPrefix)
                .build();

            // Create a list of object keys.
            ListObjectsV2Response v2Response = s3Client.listObjectsV2(v2Request);
            List<S3Object> objects = v2Response.contents();
            for (S3Object object: objects) {
                key = object.key();
                objectKeys.add(key);
            }

            return objectKeys;
        } catch ( PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    private static void printExportJobStatus(PinpointClient pinpointClient,
                                         String applicationId,
                                         String jobId) {

        GetExportJobResponse getExportJobResult;
        String status;

        try {
            // Checks the job status until the job completes or fails.
            GetExportJobRequest exportJobRequest = GetExportJobRequest.builder()

```

```
.jobId(jobId)
.applicationId(applicationId)
.build();

do {
    getExportJobResult = pinpointClient.getExportJob(exportJobRequest);
    status =
getExportJobResult.exportJobResponse().jobStatus().toString().toUpperCase();
    System.out.format("Export job %s . . \n", status);
    TimeUnit.SECONDS.sleep(3);

} while (!status.equals("COMPLETED") && !status.equals("FAILED"));

if (status.equals("COMPLETED")) {
    System.out.println("Finished exporting endpoints.");
} else {
    System.err.println("Failed to export endpoints.");
    System.exit(1);
}

} catch (PinpointException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Download files from an Amazon S3 bucket and write them to the path location.
public static void downloadFromS3(S3Client s3Client, String path, String
s3BucketName, List<String> objectKeys) {

    String newPath;
    try {
        for (String key : objectKeys) {
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                .bucket(s3BucketName)
                .key(key)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
Date());
            newPath = path + fileSuffix+".gz";
            File myFile = new File(newPath);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
        }
        System.out.println("Download finished.");
    } catch (S3Exception | NullPointerException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateExportJob](#) in *AWS SDK for Java 2.x API Reference*.

## Get endpoints

The following code example shows how to get endpoints.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void lookupPinpointEndpoint(PinpointClient pinpoint, String appId,
String endpoint) {

    try {
        GetEndpointRequest appRequest = GetEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpoint)
            .build();

        GetEndpointResponse result = pinpoint.getEndpoint(appRequest);
        EndpointResponse endResponse = result.endpointResponse();

        // Uses the Google Gson library to pretty print the endpoint JSON.
        Gson gson = new GsonBuilder()
            .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
            .setPrettyPrinting()
            .create();

        String endpointJson = gson.toJson(endResponse);
        System.out.println(endpointJson);

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}
```

- For API details, see [GetEndpoint](#) in *AWS SDK for Java 2.x API Reference*.

## Import a segment

The following code example shows how to import a segment.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Import a segment.

```
public static ImportJobResponse createImportSegment(PinpointClient client,
                                                    String appId,
                                                    String bucket,
                                                    String key,
                                                    String roleArn) {

    try {
        ImportJobRequest importRequest = ImportJobRequest.builder()
            .defineSegment(true)
            .registerEndpoints(true)
            .roleArn(roleArn)
            .format(Format.JSON)
```

```
.s3Url("s3://" + bucket + "/" + key)
.build();

CreateImportJobRequest jobRequest = CreateImportJobRequest.builder()
.importJobRequest(importRequest)
.applicationId(appId)
.build();

CreateImportJobResponse jobResponse = client.createImportJob(jobRequest);
return jobResponse.importJobResponse();

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}
```

- For API details, see [CreateImportJob](#) in *AWS SDK for Java 2.x API Reference*.

## List endpoints

The following code example shows how to list endpoints.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllEndpoints(PinpointClient pinpoint,
                                    String applicationId,
                                    String userId) {

    try {
        GetUserEndpointsRequest endpointsRequest =
GetUserEndpointsRequest.builder()
        .userId(userId)
        .applicationId(applicationId)
        .build();

        GetUserEndpointsResponse response =
pinpoint.getUserEndpoints(endpointsRequest);
        List<EndpointResponse> endpoints = response.endpointsResponse().item();

        // Display the results.
        for (EndpointResponse endpoint: endpoints) {
            System.out.println("The channel type is: "+endpoint.channelType());
            System.out.println("The address is " + endpoint.address());
        }

    } catch ( PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see  [GetUserEndpoints](#) in *AWS SDK for Java 2.x API Reference*.

## List segments

The following code example shows how to list segments.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List segments.

```
public static void listSegs( PinpointClient pinpoint, String appId) {  
  
    try {  
        GetSegmentsRequest request = GetSegmentsRequest.builder()  
            .applicationId(appId)  
            .build();  
  
        GetSegmentsResponse response = pinpoint.getSegments(request);  
        List<SegmentResponse> segments = response.segmentsResponse().item();  
        for(SegmentResponse segment: segments) {  
            System.out.println("Segment " + segment.id() + " " + segment.name() +  
" " + segment.lastModifiedDate());  
        }  
  
    } catch ( PinpointException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [GetSegments](#) in *AWS SDK for Java 2.x API Reference*.

## Send email and text messages

The following code example shows how to send email and text messages with Amazon Pinpoint.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Send an email message.

```
public static void sendEmail(PinpointClient pinpoint,  
                            String subject,  
                            String appId,  
                            String senderAddress,  
                            String toAddress) {  
  
    try {  
        Map<String,AddressConfiguration> addressMap = new HashMap<>();  
        AddressConfiguration configuration = AddressConfiguration.builder()  
            .channelType(ChannelType.EMAIL)  
            .build();  
  
        addressMap.put(toAddress, configuration);  
        SimpleEmailPart emailPart = SimpleEmailPart.builder()  
            .data(htmlBody)
```

```
.charset(charset)
.build() ;

SimpleEmailPart subjectPart = SimpleEmailPart.builder()
.data(subject)
.charset(charset)
.build() ;

SimpleEmail simpleEmail = SimpleEmail.builder()
.htmlPart(emailPart)
.subject(subjectPart)
.build();

EmailMessage emailMessage = EmailMessage.builder()
.body(htmlBody)
.fromAddress(senderAddress)
.simpleEmail(simpleEmail)
.build();

DirectMessageConfiguration directMessageConfiguration =
DirectMessageConfiguration.builder()
.emailMessage(emailMessage)
.build();

MessageRequest messageRequest = MessageRequest.builder()
.addresses(addressMap)
.messageConfiguration(directMessageConfiguration)
.build();

SendMessagesRequest messagesRequest = SendMessagesRequest.builder()
.applicationId(appId)
.messageRequest(messageRequest)
.build();

pinpoint.sendMessages(messagesRequest);

} catch (PinpointException e) {
System.err.println(e.awsErrorDetails().errorMessage());
System.exit(1);
}
}
```

Send an SMS message.

```
public static void sendSMSMessage(PinpointClient pinpoint, String message, String
appId, String originationNumber, String destinationNumber) {

try {
Map<String, AddressConfiguration> addressMap = new HashMap<String,
AddressConfiguration>();
AddressConfiguration addConfig = AddressConfiguration.builder()
.channelType(ChannelType.SMS)
.build();

addressMap.put(destinationNumber, addConfig);
SMSMessage smsMessage = SMSMessage.builder()
.body(message)
.messageType(messageType)
.originationNumber(originationNumber)
.senderId(senderId)
.keyword(registeredKeyword)
.build();

// Create a DirectMessageConfiguration object.
```

```
DirectMessageConfiguration direct = DirectMessageConfiguration.builder()  
.smsMessage(smsMessage)  
.build();  
  
MessageRequest msgReq = MessageRequest.builder()  
.addresses(addressMap)  
.messageConfiguration(direct)  
.build();  
  
// create a SendMessagesRequest object  
SendMessagesRequest request = SendMessagesRequest.builder()  
.applicationId(appId)  
.messageRequest(msgReq)  
.build();  
  
SendMessagesResponse response= pinpoint.sendMessages(request);  
MessageResponse msg1 = response.messageResponse();  
Map map1 = msg1.result();  
  
//Write out the result of sendMessage.  
map1.forEach((k, v) -> System.out.println((k + ":" + v)));  
  
} catch (PinpointException e) {  
    System.err.println(e.awsErrorDetails().errorMessage());  
    System.exit(1);  
}  
}
```

#### Send batch SMS messages.

```
public static void sendSMSMessage(PinpointClient pinpoint, String message, String  
appId, String originationNumber, String destinationNumber, String destinationNumber1)  
{  
    try {  
        Map<String, AddressConfiguration> addressMap = new HashMap<String,  
AddressConfiguration>();  
        AddressConfiguration addConfig = AddressConfiguration.builder()  
.channelType(ChannelType.SMS)  
.build();  
  
        // Add an entry to the Map object for each number to whom you want to send  
a message.  
        addressMap.put(destinationNumber, addConfig);  
        addressMap.put(destinationNumber1, addConfig);  
        SMSMessage smsMessage = SMSMessage.builder()  
.body(message)  
.messageType(messageType)  
.originationNumber(originationNumber)  
.senderId(senderId)  
.keyword(registeredKeyword)  
.build();  
  
        // Create a DirectMessageConfiguration object.  
        DirectMessageConfiguration direct = DirectMessageConfiguration.builder()  
.smsMessage(smsMessage)  
.build();  
  
        MessageRequest msgReq = MessageRequest.builder()  
.addresses(addressMap)  
.messageConfiguration(direct)  
.build();  
  
        // Create a SendMessagesRequest object.  
        SendMessagesRequest request = SendMessagesRequest.builder()
```

```
.applicationId(appId)
.messageRequest(msgReq)
.build();

SendMessagesResponse response= pinpoint.sendMessages(request);
MessageResponse msg1 = response.messageResponse();
Map map1 = msg1.result();

// Write out the result of sendMessage.
map1.forEach((k, v) -> System.out.println((k + ":" + v)));

} catch (PinpointException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [SendMessages](#) in *AWS SDK for Java 2.x API Reference*.

## Update an endpoint

The following code example shows how to update an endpoint.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static EndpointResponse createEndpoint(PinpointClient client, String appId)
{
    String endpointId = UUID.randomUUID().toString();
    System.out.println("Endpoint ID: " + endpointId);

    try {
        EndpointRequest endpointRequest = createEndpointRequestData();
        UpdateEndpointRequest updateEndpointRequest =
UpdateEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpointId)
            .endpointRequest(endpointRequest)
            .build();

        UpdateEndpointResponse updateEndpointResponse =
client.updateEndpoint(updateEndpointRequest);
        System.out.println("Update Endpoint Response: " +
updateEndpointResponse.messageBody());

        GetEndpointRequest getEndpointRequest = GetEndpointRequest.builder()
            .applicationId(appId)
            .endpointId(endpointId)
            .build();

        GetEndpointResponse getEndpointResponse =
client.getEndpoint(getEndpointRequest);
        System.out.println(getEndpointResponse.endpointResponse().address());
        System.out.println(getEndpointResponse.endpointResponse().channelType());
        System.out.println(getEndpointResponse.endpointResponse().applicationId());

System.out.println(getEndpointResponse.endpointResponse().endpointStatus());
        System.out.println(getEndpointResponse.endpointResponse().requestId());
    }
}
```

```
        System.out.println(getEndpointResponse.endpointResponse().user());

        return getEndpointResponse.endpointResponse();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

private static EndpointRequest createEndpointRequestData() {

    try {
        List<String> favoriteTeams = new ArrayList<>();
        favoriteTeams.add("Lakers");
        favoriteTeams.add("Warriors");
        HashMap<String, List<String>> customAttributes = new HashMap<>();
        customAttributes.put("team", favoriteTeams);

        EndpointDemographic demographic = EndpointDemographic.builder()
            .appVersion("1.0")
            .make("apple")
            .model("iPhone")
            .modelVersion("7")
            .platform("ios")
            .platformVersion("10.1.1")
            .timezone("America/Los_Angeles")
            .build();

        EndpointLocation location = EndpointLocation.builder()
            .city("Los Angeles")
            .country("US")
            .latitude(34.0)
            .longitude(-118.2)
            .postalCode("90068")
            .region("CA")
            .build();

        Map<String,Double> metrics = new HashMap<>();
        metrics.put("health", 100.00);
        metrics.put("luck", 75.00);

        EndpointUser user = EndpointUser.builder()
            .userId(UUID.randomUUID().toString())
            .build();

        DateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm'Z'"); // Quoted
        "Z" to indicate UTC, no timezone offset
        String nowAsISO = df.format(new Date());

        return EndpointRequest.builder()
            .address(UUID.randomUUID().toString())
            .attributes(customAttributes)
            .channelType("APNS")
            .demographic(demographic)
            .effectiveDate(nowAsISO)
            .location(location)
            .metrics(metrics)
            .optOut("NONE")
            .requestId(UUID.randomUUID().toString())
            .user(user)
            .build();

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
    return null;
}
```

- For API details, see [UpdateEndpoint in AWS SDK for Java 2.x API Reference](#).

## Update channels

The following code example shows how to update channels.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
private static SMSChannelResponse getSMSChannel(PinpointClient client, String appId) {

    try {
        GetSmsChannelRequest request = GetSmsChannelRequest.builder()
            .applicationId(appId)
            .build();

        SMSChannelResponse response =
client.getSMSChannel(request).smsChannelResponse();
        System.out.println("Channel state is " + response.enabled());
        return response;
    } catch ( PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

private static void toggleSMSChannel(PinpointClient client, String appId,
SMSChannelResponse getResponse) {
    boolean enabled = !getResponse.enabled();

    try {
        SMSChannelRequest request = SMSChannelRequest.builder()
            .enabled(enabled)
            .build();

        UpdateSMSChannelRequest updateRequest = UpdateSMSChannelRequest.builder()
            .smsChannelRequest(request)
            .applicationId(appId)
            .build();

        UpdateSMSChannelResponse result = client.updateSMSChannel(updateRequest);
        System.out.println("Channel state: " +
result.smsChannelResponse().enabled());

    } catch ( PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [GetSmsChannel](#) in *AWS SDK for Java 2.x API Reference*.

## Amazon Pinpoint SMS and Voice API examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Pinpoint SMS and Voice API.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3358\)](#)

## Actions

### Send a voice message with Amazon Pinpoint SMS and Voice API

The following code example shows how to send a voice message with Amazon Pinpoint SMS and Voice API.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
originationNumber, String destinationNumber ) {

    try {
        SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
            .languageCode(languageCode)
            .text(ssmlMessage)
            .voiceId(voiceName)
            .build();

        VoiceMessageContent content = VoiceMessageContent.builder()
            .ssmlMessage(ssmlMessageType)
            .build();

        SendVoiceMessageRequest voiceMessageRequest =
SendVoiceMessageRequest.builder()
            .destinationPhoneNumber(destinationNumber)
            .originationPhoneNumber(originationNumber)
            .content(content)
            .build();

        client.sendVoiceMessage(voiceMessageRequest);
        System.out.println("The message was sent successfully.");

    } catch (PinpointSmsVoiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SendVoiceMessage](#) in *AWS SDK for Java 2.x API Reference*.

## Amazon RDS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Relational Database Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3359\)](#)
- [Scenarios \(p. 3368\)](#)

## Actions

### Create a DB instance

The following code example shows how to create an Amazon RDS DB instance and wait for it to become available.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createSnapshot(RdsClient rdsClient, String dbInstanceIdentifier,
String dbSnapshotIdentifier) {

    try {
        CreateDbSnapshotRequest snapshotRequest = CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.print("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

### Create a DB parameter group

The following code example shows how to create an Amazon RDS DB parameter group.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createDBParameterGroup(RdsClient rdsClient, String dbGroupName,
String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
        .dbParameterGroupName(dbGroupName)
        .dbParameterGroupFamily(dbParameterGroupFamily)
        .description("Created by using the AWS SDK for Java")
        .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateDBParameterGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Create a snapshot of a DB instance

The following code example shows how to create a snapshot of an Amazon RDS DB instance.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String dbInstanceIdentifier,
String dbSnapshotIdentifier) {
    try {
        CreateDbSnapshotRequest snapshotRequest = CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateDBSnapshot](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a DB instance

The following code example shows how to delete an Amazon RDS DB instance.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteDatabaseInstance( RdsClient rdsClient, String dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .deleteAutomatedBackups(true)
        .skipFinalSnapshot(true)
        .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.print("The status of the database is " +
response.dbInstance().dbInstanceState());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a DB parameter group

The following code example shows how to delete an Amazon RDS DB parameter group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Delete the parameter group after database has been deleted.
// An exception is thrown if you attempt to delete the para group while database
exists.
public static void deleteParaGroup( RdsClient rdsClient, String dbGroupName, String
dbARN) throws InterruptedException {
    try {
        boolean isDataDel = false;
        boolean didFind;
        String instanceARN ;

        // Make sure that the database has been deleted.
        while (!isDataDel) {
            DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
```

```
List<DBInstance> instanceList = response.dbInstances();
int listSize = instanceList.size();
isDataDel = false ;
didFind = false;
int index = 1;
for (DBInstance instance: instanceList) {
    instanceARN = instance.dbInstanceArn();
    if (instanceARN.compareTo(dbARN) == 0) {
        System.out.println(dbARN + " still exists");
        didFind = true ;
    }
    if ((index == listSize) && (!didFind)) {
        // Went through the entire list and did not find the database
        ARN.
        isDataDel = true;
    }
    Thread.sleep(sleepTime * 1000);
    index++;
}
}

// Delete the para group.
DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
    .dbParameterGroupName(dbGroupName)
    .build();

rdsClient.deleteDBParameterGroup(parameterGroupRequest);
System.out.println(dbGroupName +" was deleted.");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- For API details, see [DeleteDBParameterGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Describe DB instances

The following code example shows how to describe Amazon RDS DB instances.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeInstances(RdsClient rdsClient) {

    try {
        DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
        List<DBInstance> instanceList = response.dbInstances();
        for (DBInstance instance: instanceList) {
            System.out.println("Instance ARN is: "+instance.dbInstanceArn());
            System.out.println("The Engine is " +instance.engine());
            System.out.println("Connection endpoint is"
+instance.endpoint().address());
        }
    } catch (RdsException e) {
```

```
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for Java 2.x API Reference*.

## Describe DB parameter groups

The following code example shows how to describe Amazon RDS DB parameter groups.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
        .dbParameterGroupName(dbGroupName)
        .maxRecords(20)
        .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
        List<DBParameterGroup> groups = response.dbParameterGroups();
        for (DBParameterGroup group: groups) {
            System.out.println("The group name is "+group.dbParameterGroupName());
            System.out.println("The group description is "+group.description());
        }
    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBParameterGroups](#) in *AWS SDK for Java 2.x API Reference*.

## Describe database engine versions

The following code example shows how to describe Amazon RDS database engine versions.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeDBEngines( RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
        .defaultOnly(true)
```

```
.engine("mysql")
.maxRecords(20)
.build();

DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
List<DBEngineVersion> engines = response.dbEngineVersions();

// Get all DBEngineVersion objects.
for (DBEngineVersion engine0b: engines) {
    System.out.println("The name of the DB parameter group family for the
database engine is "+engine0b.dbParameterGroupFamily());
    System.out.println("The name of the database engine
"+engine0b.engine());
    System.out.println("The version number of the database engine
"+engine0b.engineVersion());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- For API details, see [DescribeDBEngineVersions](#) in *AWS SDK for Java 2.x API Reference*.

## Describe options for DB instances

The following code example shows how to describe options for Amazon RDS DB instances.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
    try {
        DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
        .dbParameterGroupFamily(dbParameterGroupFamily)
        .engine("mysql")
        .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
        List<DBEngineVersion> dbEngines = response.dbEngineVersions();
        for (DBEngineVersion dbEngine: dbEngines) {
            System.out.println("The engine version is " +dbEngine.engineVersion());
            System.out.println("The engine description is "
+dbEngine.dbEngineDescription());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeOrderableDBInstanceState](#) in *AWS SDK for Java 2.x API Reference*.

## Describe parameters in a DB parameter group

The following code example shows how to describe parameters in an Amazon RDS DB parameter group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName, int
flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .source("user")
                .build();
        }

        DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para: dbParameters) {
            // Only print out information about either auto_increment_offset or
auto_increment_increment.
            paraName = para.parameterName();
            if ( (paraName.compareTo("auto_increment_offset") ==0) ||
(paraName.compareTo("auto_increment_increment ") ==0)) {
                System.out.println("**** The parameter name is " + paraName);
                System.out.println("**** The parameter value is " +
para.parameterValue());
                System.out.println("**** The parameter data type is " +
para.dataType());
                System.out.println("**** The parameter description is " +
para.description());
                System.out.println("**** The parameter allowed values is " +
para.allowedValues());
            }
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeDBParameters](#) in *AWS SDK for Java 2.x API Reference*.

## Modify a DB instance

The following code example shows how to modify an Amazon RDS DB instance.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void updateInstance(RdsClient rdsClient, String dbInstanceIdentifier,
String masterUserPassword) {

    try {
        // For a demo - modify the DB instance by modifying the master password.
        ModifyDbInstanceRequest modifyDbInstanceRequest =
ModifyDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .publiclyAccessible(true)
            .masterUserPassword(masterUserPassword)
            .build();

        ModifyDbInstanceResponse instanceResponse =
rdsClient.modifyDBInstance(modifyDbInstanceRequest);
        System.out.print("The ARN of the modified database is: "
+instanceResponse.dbInstance().dbInstanceArn());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [ModifyDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

## Reboot a DB instance

The following code example shows how to reboot an Amazon RDS DB instance.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void rebootInstance(RdsClient rdsClient, String
dbInstanceIdentifier ) {

    try {
        RebootDbInstanceRequest rebootDbInstanceRequest =
RebootDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        RebootDbInstanceResponse instanceResponse =
rdsClient.rebootDBInstance(rebootDbInstanceRequest);
        System.out.print("The database " +
instanceResponse.dbInstance().dbInstanceArn() +" was rebooted");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [RebootDBInstance](#) in *AWS SDK for Java 2.x API Reference*.

## Retrieve attributes

The following code example shows how to retrieve attributes that belong to an Amazon RDS account.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getAccountAttributes(RdsClient rdsClient) {  
  
    try {  
        DescribeAccountAttributesResponse response =  
rdsClient.describeAccountAttributes();  
        List<AccountQuota> quotasList = response.accountQuotas();  
        for (AccountQuota quotas: quotasList) {  
            System.out.println("Name is: " +quotas.accountQuotaName());  
            System.out.println("Max value is " +quotas.max());  
        }  
  
    } catch (RdsException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DescribeAccountAttributes](#) in *AWS SDK for Java 2.x API Reference*.

## Update parameters in a DB parameter group

The following code example shows how to update parameters in an Amazon RDS DB parameter group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Modify auto_increment_offset and auto_increment_increment parameters.  
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {  
    try {  
        Parameter parameter1 = Parameter.builder()  
            .parameterName("auto_increment_offset")  
            .applyMethod("immediate")  
            .parameterValue("5")  
            .build();  
  
        List<Parameter> paraList = new ArrayList<>();  
        paraList.add(parameter1);  
        ModifyDbParameterGroupRequest groupRequest =  
ModifyDbParameterGroupRequest.builder()  
            .dBParameterGroupName(dbGroupName)  
            .parameters(paraList)  
            .build();
```

```
        ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
System.out.println("The parameter group "+ response.dbParameterGroupName()
+" was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [ModifyDBParameterGroup](#) in *AWS SDK for Java 2.x API Reference*.

## Scenarios

### Get started with DB instances

The following code example shows how to:

- Create a custom DB parameter group and set parameter values.
- Create a DB instance that's configured to use the parameter group. The DB instance also contains a database.
- Take a snapshot of the instance.
- Delete the instance and parameter group.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run multiple operations.

```
/**
 * Before running this Java (v2) code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Returns a list of the available DB engines.
 * 2. Selects an engine family and create a custom DB parameter group.
 * 3. Gets the parameter groups.
 * 4. Gets parameters in the group.
 * 5. Modifies the auto_increment_offset parameter.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions.
 * 8. Gets a list of micro instance classes available for the selected engine.
 * 9. Creates an RDS database instance that contains a MySQL database and uses the
 *    parameter group.
 * 10. Waits for the DB instance to be ready and prints out the connection endpoint
 *     value.
 * 11. Creates a snapshot of the DB instance.
 * 12. Waits for an RDS DB snapshot to be ready.
 * 13. Deletes the RDS DB instance.
```

```
* 14. Deletes the parameter group.  
*/  
public class RDSScenario {  
  
    public static long sleepTime = 20;  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    public static void main(String[] args) throws InterruptedException {  
  
        final String usage = "\n" +  
            "Usage:\n" +  
            "  <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier> <dbName>  
<masterUsername> <masterUserPassword> <dbSnapshotIdentifier>\n\n" +  
            "Where:\n" +  
            "  dbGroupName - The database group name. \n"+  
            "  dbParameterGroupFamily - The database parameter group name (for  
example, mysql8.0).\n"+  
            "  dbInstanceIdentifier - The database instance identifier \n"+  
            "  dbName - The database name. \n"+  
            "  masterUsername - The master user name. \n"+  
            "  masterUserPassword - The password that corresponds to the master user  
name. \n"+  
            "  dbSnapshotIdentifier - The snapshot identifier. \n" ;  
  
        if (args.length != 7) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String dbGroupName = args[0];  
        String dbParameterGroupFamily = args[1];  
        String dbInstanceIdentifier = args[2];  
        String dbName = args[3];  
        String masterUsername = args[4];  
        String masterUserPassword = args[5];  
        String dbSnapshotIdentifier = args[6];  
  
        Region region = Region.US_WEST_2;  
        RdsClient rdsClient = RdsClient.builder()  
            .region(region)  
            .credentialsProvider(ProfileCredentialsProvider.create())  
            .build();  
        System.out.println(DASHES);  
        System.out.println("Welcome to the Amazon RDS example scenario.");  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("1. Return a list of the available DB engines");  
        describeDBEngines(rdsClient);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("2. Create a custom parameter group");  
        createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("3. Get the parameter group");  
        describeDbParameterGroups(rdsClient, dbGroupName);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("4. Get the parameters in the group");  
        describeDbParameters(rdsClient, dbGroupName, 0);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);
```

```
System.out.println("5. Modify the auto_increment_offset parameter");
modifyDBParas(rdsClient, dbGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Display the updated value");
describeDbParameters(rdsClient, dbGroupName, -1);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of allowed engine versions");
getAllowedEngines(rdsClient, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Get a list of micro instance classes available for the
selected engine") ;
getMicroInstances(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Create an RDS database instance that contains a MySQL
database and uses the parameter group");
String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
dbInstanceIdentifier, dbName, masterUsername, masterUserPassword);
System.out.println("The ARN of the new database is "+dbARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Wait for DB instance to be ready" );
waitForInstanceReady(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Create a snapshot of the DB instance");
createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Wait for DB snapshot to be ready" );
waitForSnapshotReady(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Delete the DB instance" );
deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Delete the parameter group");
deleteParaGroup(rdsClient, dbGroupName, dbARN);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The Scenario has successfully completed." );
System.out.println(DASHES);

rdsClient.close();
}

// Delete the parameter group after database has been deleted.
// An exception is thrown if you attempt to delete the para group while database
exists.
public static void deleteParaGroup( RdsClient rdsClient, String dbGroupName, String
dbARN) throws InterruptedException {
try {
```

```
boolean isDataDel = false;
boolean didFind;
String instanceARN ;

// Make sure that the database has been deleted.
while (!isDataDel) {
    DescribeDbInstancesResponse response = rdsClient.describeDBInstances();
    List<DBInstance> instanceList = response.dbInstances();
    int listSize = instanceList.size();
    isDataDel = false ;
    didFind = false;
    int index = 1;
    for (DBInstance instance: instanceList) {
        instanceARN = instance.dbInstanceArn();
        if (instanceARN.compareTo(dbARN) == 0) {
            System.out.println(dbARN + " still exists");
            didFind = true ;
        }
        if ((index == listSize) && (!didFind)) {
            // Went through the entire list and did not find the database
            ARN.
            isDataDel = true;
        }
        Thread.sleep(sleepTime * 1000);
        index++;
    }
}

// Delete the para group.
DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
    .dbParameterGroupName(dbGroupName)
    .build();

rdsClient.deleteDBParameterGroup(parameterGroupRequest);
System.out.println(dbGroupName +" was deleted.");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Delete the DB instance.
public static void deleteDatabaseInstance( RdsClient rdsClient, String
dbInstanceIdentifier) {
try {
    DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
    .dbInstanceIdentifier(dbInstanceIdentifier)
    .deleteAutomatedBackups(true)
    .skipFinalSnapshot(true)
    .build();

    DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
    System.out.print("The status of the database is " +
response.dbInstance().dbInstanceState());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Waits until the snapshot instance is available.
```

```
public static void waitForSnapshotReady(RdsClient rdsClient, String dbInstanceIdentifier, String dbSnapshotIdentifier) {
    try {
        boolean snapshotReady = false;
        String snapshotReadyStr;
        System.out.println("Waiting for the snapshot to become available.");

        DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        while (!snapshotReady) {
            DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
            List<DBSnapshot> snapshotList = response.dbSnapshots();
            for (DBSnapshot snapshot : snapshotList) {
                snapshotReadyStr = snapshot.status();
                if (snapshotReadyStr.contains("available")) {
                    snapshotReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }

        System.out.println("The Snapshot is available!");
    } catch (RdsException | InterruptedException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Create an Amazon RDS snapshot.
public static void createSnapshot(RdsClient rdsClient, String dbInstanceIdentifier,
String dbSnapshotIdentifier) {
    try {
        CreateDbSnapshotRequest snapshotRequest = CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");

    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
```

```
.build();

String endpoint="";
while (!instanceReady) {
    DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
    List<DBInstance> instanceList = response.dbInstances();
    for (DBInstance instance : instanceList) {
        instanceReadyStr = instance.dbInstanceState();
        if (instanceReadyStr.contains("available")) {
            endpoint = instance.endpoint().address();
            instanceReady = true;
        } else {
            System.out.print(".");
            Thread.sleep(sleepTime * 1000);
        }
    }
    System.out.println("Database instance is available! The connection endpoint
is "+ endpoint);
} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Create a database instance and return the ARN of the database.
public static String createDatabaseInstance(RdsClient rdsClient,
                                             String dbGroupName,
                                             String dbInstanceIdentifier,
                                             String dbName,
                                             String masterUsername,
                                             String masterUserPassword) {

try {
    CreateDbInstanceRequest instanceRequest = CreateDbInstanceRequest.builder()
        .dbInstanceIdentifier(dbInstanceIdentifier)
        .allocatedStorage(100)
        .dbName(dbName)
        .dbParameterGroupName(dbGroupName)
        .engine("mysql")
        .dbInstanceClass("db.m4.large")
        .engineVersion("8.0")
        .storageType("standard")
        .masterUsername(masterUsername)
        .masterUserPassword(masterUserPassword)
        .build();

    CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
    System.out.print("The status is " +
response.dbInstance().dbInstanceState());
    return response.dbInstance().dbInstanceArn();

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

return "";
}

// Get a list of micro instances.
public static void getMicroInstances(RdsClient rdsClient) {
try {
```

```
DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest.builder()
    .engine("mysql")
    .build();

DescribeOrderableDbInstanceOptionsResponse response =
rdsClient.describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
List<OrderableDBInstanceState> orderableDBInstances =
response.orderableDBInstanceState();
for (OrderableDBInstanceState dbInstanceState: orderableDBInstances) {
    System.out.println("The engine version is "
+dbInstanceState.engineVersion());
    System.out.println("The engine description is "
+dbInstanceState.engine());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Get a list of allowed engine versions.
public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
try {
    DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
    .dbParameterGroupFamily(dbParameterGroupFamily)
    .engine("mysql")
    .build();

DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
List<DBEngineVersion> dbEngines = response.dbEngineVersions();
for (DBEngineVersion dbEngine: dbEngines) {
    System.out.println("The engine version is " +dbEngine.engineVersion());
    System.out.println("The engine description is "
+dbEngine.dbEngineDescription());
}

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParams(RdsClient rdsClient, String dbGroupName) {
try {
    Parameter parameter1 = Parameter.builder()
        .parameterName("auto_increment_offset")
        .applyMethod("immediate")
        .parameterValue("5")
        .build();

List<Parameter> paraList = new ArrayList<>();
paraList.add(parameter1);
ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
    .dbParameterGroupName(dbGroupName)
    .parameters(paraList)
    .build();

ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
```

```

        System.out.println("The parameter group "+ response.dbParameterGroupName()
+" was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName, int
flag) {
    try {
        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .source("user")
                .build();
        }

        DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para: dbParameters) {
            // Only print out information about either auto_increment_offset or
auto_increment_increment.
            paraName = para.parameterName();
            if ( (paraName.compareTo("auto_increment_offset") ==0) ||
(paraName.compareTo("auto_increment_increment ") ==0)) {
                System.out.println("**** The parameter name is " + paraName);
                System.out.println("**** The parameter value is " +
para.parameterValue());
                System.out.println("**** The parameter data type is " +
para.dataType());
                System.out.println("**** The parameter description is " +
para.description());
                System.out.println("**** The parameter allowed values is " +
para.allowedValues());
            }
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
    try {
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .maxRecords(20)
            .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
        List<DBParameterGroup> groups = response.dbParameterGroups();
        for (DBParameterGroup group: groups) {

```

```

        System.out.println("The group name is "+group.dbParameterGroupName());
        System.out.println("The group description is "+group.description());
    }

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}

}

public static void createDBParameterGroup(RdsClient rdsClient, String dbGroupName,
String dbParameterGroupFamily) {
    try {
        CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .dbParameterGroupFamily(dbParameterGroupFamily)
            .description("Created by using the AWS SDK for Java")
            .build();

        CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
        System.out.println("The group name is "+
response.dbParameterGroup().dbParameterGroupName());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeDBEngines( RdsClient rdsClient) {
    try {
        DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
            .defaultOnly(true)
            .engine("mysql")
            .maxRecords(20)
            .build();

        DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
        List<DBEngineVersion> engines = response.dbEngineVersions();

        // Get all DBEngineVersion objects.
        for (DBEngineVersion engine0b: engines) {
            System.out.println("The name of the DB parameter group family for the
database engine is "+engine0b.dbParameterGroupFamily());
            System.out.println("The name of the database engine
"+engine0b.engine());
            System.out.println("The version number of the database engine
"+engine0b.engineVersion());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}

```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
    - [CreateDBInstance](#)

- [CreateDBParameterGroup](#)
- [CreateDBSnapshot](#)
- [DeleteDBInstance](#)
- [DeleteDBParameterGroup](#)
- [DescribeDBEngineVersions](#)
- [DescribeDBInstances](#)
- [DescribeDBParameterGroups](#)
- [DescribeDBParameters](#)
- [DescribeDBSnapshots](#)
- [DescribeOrderableDBInstanceOptions](#)
- [ModifyDBParameterGroup](#)

## Amazon Redshift examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Redshift.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3377\)](#)

## Actions

### Create a cluster

The following code example shows how to create an Amazon Redshift cluster.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the cluster.

```
public static void createCluster(RedshiftClient redshiftClient, String clusterId,
String masterUsername, String masterUserPassword ) {

    try {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername) // set the user name here
            .masterUserPassword(masterUserPassword) // set the user password here
            .nodeType("ds2.xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        CreateClusterResponse clusterResponse =
redshiftClient.createCluster(clusterRequest);
        System.out.println("Created cluster " +
clusterResponse.cluster().clusterIdentifier());
    }
}
```

```
        } catch (RedshiftException e) {
            System.err.println(e.getMessage());
            System.exit(1);
    }
```

- For API details, see [CreateCluster](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a cluster

The following code example shows how to delete an Amazon Redshift cluster.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete the cluster.

```
public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
clusterId) {

    try {
        DeleteClusterRequest deleteClusterRequest = DeleteClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .skipFinalClusterSnapshot(true)
            .build();

        DeleteClusterResponse response =
redshiftClient.deleteCluster(deleteClusterRequest);
        System.out.println("The status is "+response.cluster().clusterStatus());

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteCluster](#) in *AWS SDK for Java 2.x API Reference*.

## Describe your clusters

The following code example shows how to describe your Amazon Redshift clusters.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Describe the cluster.

```
public static void describeRedshiftClusters(RedshiftClient redshiftClient) {

    try {
        DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters();
```

```
        List<Cluster> clusterList = clusterResponse.clusters();
        for (Cluster cluster: clusterList) {
            System.out.println("Cluster database name is: "+cluster.dbName());
            System.out.println("Cluster status is: "+cluster.clusterStatus());
        }

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeClusters in AWS SDK for Java 2.x API Reference](#).

## Modify a cluster

The following code example shows how to modify an Amazon Redshift cluster.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Modify a cluster.

```
public static void modifyCluster(RedshiftClient redshiftClient, String clusterId) {

    try {
        ModifyClusterRequest modifyClusterRequest = ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();

        ModifyClusterResponse clusterResponse =
        redshiftClient.modifyCluster(modifyClusterRequest);
        System.out.println("The modified cluster was successfully modified and
has "+ clusterResponse.cluster().preferredMaintenanceWindow() +" as the maintenance
window");

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [ModifyCluster in AWS SDK for Java 2.x API Reference](#).

## Amazon Rekognition examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Rekognition.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Topics

- [Actions \(p. 3380\)](#)
- [Scenarios \(p. 3391\)](#)

## Actions

### Compare faces in an image against a reference image

The following code example shows how to compare faces in an image against a reference image with Amazon Rekognition.

For more information, see [Comparing faces in images](#).

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void compareTwoFaces(RekognitionClient rekClient, Float similarityThreshold, String sourceImage, String targetImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        InputStream tarStream = new FileInputStream(targetImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        SdkBytes targetBytes = SdkBytes.fromInputStream(tarStream);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        Image tarImage = Image.builder()
            .bytes(targetBytes)
            .build();

        CompareFacesRequest facesRequest = CompareFacesRequest.builder()
            .sourceImage(souImage)
            .targetImage(tarImage)
            .similarityThreshold(similarityThreshold)
            .build();

        // Compare the two images.
        CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
        List<CompareFacesMatch> faceDetails = compareFacesResult.faceMatches();
        for (CompareFacesMatch match: faceDetails){
            ComparedFace face= match.face();
            BoundingBox position = face.boundingBox();
            System.out.println("Face at " + position.left().toString()
                + " " + position.top()
                + " matches with " + face.confidence().toString()
                + "% confidence.");

        }
        List<ComparedFace> uncompered = compareFacesResult.unmatchedFaces();
        System.out.println("There was " + uncompered.size() + " face(s) that did
not match");
        System.out.println("Source image rotation: " +
compareFacesResult.sourceImageOrientationCorrection());
        System.out.println("target image rotation: " +
compareFacesResult.targetImageOrientationCorrection());
```

```
        } catch(RekognitionException | FileNotFoundException e) {
            System.out.println("Failed to load source image " + sourceImage);
            System.exit(1);
    }
}
```

- For API details, see [CompareFaces](#) in *AWS SDK for Java 2.x API Reference*.

## Create a collection

The following code example shows how to create an Amazon Rekognition collection.

For more information, see [Creating a collection](#).

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createMyCollection(RekognitionClient rekClient, String
collectionId) {

    try {
        CreateCollectionRequest collectionRequest =
CreateCollectionRequest.builder()
            .collectionId(collectionId)
            .build();

        CreateCollectionResponse collectionResponse =
rekClient.createCollection(collectionRequest);
        System.out.println("CollectionArn: " + collectionResponse.collectionArn());
        System.out.println("Status code: " +
collectionResponse.statusCode().toString());

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateCollection](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a collection

The following code example shows how to delete an Amazon Rekognition collection.

For more information, see [Deleting a collection](#).

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteMyCollection(RekognitionClient rekClient, String
collectionId) {
```

```
try {
    DeleteCollectionRequest deleteCollectionRequest =
DeleteCollectionRequest.builder()
    .collectionId(collectionId)
    .build();

    DeleteCollectionResponse deleteCollectionResponse =
rekClient.deleteCollection(deleteCollectionRequest);
    System.out.println(collectionId + ":" +
deleteCollectionResponse.statusCode().toString());

} catch(RekognitionException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [DeleteCollection in AWS SDK for Java 2.x API Reference](#).

## Delete faces from a collection

The following code example shows how to delete faces from an Amazon Rekognition collection.

For more information, see [Deleting faces from a collection](#).

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteFacesCollection(RekognitionClient rekClient,
                                         String collectionId,
                                         String faceId) {

    try {
        DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
            .collectionId(collectionId)
            .faceIds(faceId)
            .build();

        rekClient.deleteFaces(deleteFacesRequest);
        System.out.println("The face was deleted from the collection.");

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteFaces in AWS SDK for Java 2.x API Reference](#).

## Describe a collection

The following code example shows how to describe an Amazon Rekognition collection.

For more information, see [Describing a collection](#).

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeColl(RekognitionClient rekClient, String collectionName) {
    try {
        DescribeCollectionRequest describeCollectionRequest =
DescribeCollectionRequest.builder()
    .collectionId(collectionName)
    .build();

        DescribeCollectionResponse describeCollectionResponse =
rekClient.describeCollection(describeCollectionRequest);
        System.out.println("Collection Arn : " +
describeCollectionResponse.collectionARN());
        System.out.println("Created : " +
describeCollectionResponse.creationTimestamp().toString());

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeCollection](#) in *AWS SDK for Java 2.x API Reference*.

## Detect faces in an image

The following code example shows how to detect faces in an image with Amazon Rekognition.

For more information, see [Detecting faces in an image](#).

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectFacesinImage(RekognitionClient rekClient, String
sourceImage ) {

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectFacesRequest facesRequest = DetectFacesRequest.builder()
            .attributes(Attribute.ALL)
            .image(souImage)
            .build();

        DetectFacesResponse facesResponse = rekClient.detectFaces(facesRequest);
```

```
List<FaceDetail> faceDetails = facesResponse.faceDetails();
for (FaceDetail face : faceDetails) {
    AgeRange ageRange = face.ageRange();
    System.out.println("The detected face is estimated to be between "
        + ageRange.low().toString() + " and " +
    ageRange.high().toString()
        + " years old.");

    System.out.println("There is a smile :
"+face.smile().value().toString());
}

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [DetectFaces](#) in *AWS SDK for Java 2.x API Reference*.

## Detect labels in an image

The following code example shows how to detect labels in an image with Amazon Rekognition.

For more information, see [Detecting labels in an image](#).

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectImageLabels(RekognitionClient rekClient, String
sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
            .image(souImage)
            .maxLabels(10)
            .build();

        DetectLabelsResponse labelsResponse =
rekClient.detectLabels(detectLabelsRequest);
        List<Label> labels = labelsResponse.labels();
        System.out.println("Detected labels for the given photo");
        for (Label label: labels) {
            System.out.println(label.name() + ": " +
label.confidence().toString());
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }
```

- For API details, see [DetectLabels](#) in *AWS SDK for Java 2.x API Reference*.

## Detect moderation labels in an image

The following code example shows how to detect moderation labels in an image with Amazon Rekognition. Moderation labels identify content that may be inappropriate for some audiences.

For more information, see [Detecting inappropriate images](#).

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectModLabels(RekognitionClient rekClient, String sourceImage)
{
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectModerationLabelsRequest moderationLabelsRequest =
        DetectModerationLabelsRequest.builder()
            .image(souImage)
            .minConfidence(60F)
            .build();

        DetectModerationLabelsResponse moderationLabelsResponse =
        rekClient.detectModerationLabels(moderationLabelsRequest);
        List<ModerationLabel> labels = moderationLabelsResponse.moderationLabels();
        System.out.println("Detected labels for image");

        for (ModerationLabel label : labels) {
            System.out.println("Label: " + label.name()
                + "\n Confidence: " + label.confidence().toString() + "%"
                + "\n Parent:" + label.parentName());
        }
    } catch (RekognitionException | FileNotFoundException e) {
        e.printStackTrace();
        System.exit(1);
    }
}
```

- For API details, see [DetectModerationLabels](#) in *AWS SDK for Java 2.x API Reference*.

## Detect text in an image

The following code example shows how to detect text in an image with Amazon Rekognition.

For more information, see [Detecting text in an image](#).

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void detectTextLabels(RekognitionClient rekClient, String sourceImage) {  
  
    try {  
        InputStream sourceStream = new FileInputStream(sourceImage);  
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);  
        Image souImage = Image.builder()  
            .bytes(sourceBytes)  
            .build();  
  
        DetectTextRequest textRequest = DetectTextRequest.builder()  
            .image(souImage)  
            .build();  
  
        DetectTextResponse textResponse = rekClient.detectText(textRequest);  
        List<TextDetection> textCollection = textResponse.textDetections();  
        System.out.println("Detected lines and words");  
        for (TextDetection text: textCollection) {  
            System.out.println("Detected: " + text.detectedText());  
            System.out.println("Confidence: " + text.confidence().toString());  
            System.out.println("Id : " + text.id());  
            System.out.println("Parent Id: " + text.parentId());  
            System.out.println("Type: " + text.type());  
            System.out.println();  
        }  
    } catch (RekognitionException | FileNotFoundException e) {  
        System.out.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [DetectText](#) in *AWS SDK for Java 2.x API Reference*.

## Index faces to a collection

The following code example shows how to index faces in an image and add them to an Amazon Rekognition collection.

For more information, see [Adding faces to a collection](#).

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void addToCollection(RekognitionClient rekClient, String collectionId, String sourceImage) {  
  
    try {  
        InputStream sourceStream = new FileInputStream(sourceImage);  
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);  
        Image souImage = Image.builder()
```

```
.bytes(sourceBytes)
.build();

IndexFacesRequest facesRequest = IndexFacesRequest.builder()
.collectionId(collectionId)
.image(souImage)
.maxFaces(1)
.qualityFilter(QualityFilter.AUTO)
.detectionAttributes(Attribute.DEFAULT)
.build();

IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
System.out.println("Results for the image");
System.out.println("\n Faces indexed:");
List<FaceRecord> faceRecords = facesResponse.faceRecords();
for (FaceRecord faceRecord : faceRecords) {
    System.out.println(" Face ID: " + faceRecord.face().faceId());
    System.out.println(" Location:" +
faceRecord.faceDetail().boundingBox().toString());
}

List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
System.out.println("Faces not indexed:");
for (UnindexedFace unindexedFace : unindexedFaces) {
    System.out.println(" Location:" +
unindexedFace.faceDetail().boundingBox().toString());
    System.out.println(" Reasons:");
    for (Reason reason : unindexedFace.reasons()) {
        System.out.println("Reason: " + reason);
    }
}

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [IndexFaces](#) in *AWS SDK for Java 2.x API Reference*.

## List collections

The following code example shows how to list Amazon Rekognition collections.

For more information, see [Listing collections](#).

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllCollections(RekognitionClient rekClient) {
    try {
        ListCollectionsRequest listCollectionsRequest =
ListCollectionsRequest.builder()
        .maxResults(10)
        .build();

        ListCollectionsResponse response =
rekClient.listCollections(listCollectionsRequest);
        List<String> collectionIds = response.collectionIds();
```

```
        for (String resultId : collectionIds) {
            System.out.println(resultId);
        }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListCollections](#) in *AWS SDK for Java 2.x API Reference*.

## List faces in a collection

The following code example shows how to list faces in an Amazon Rekognition collection.

For more information, see [Listing faces in a collection](#).

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listFacesCollection(RekognitionClient rekClient, String
collectionId ) {
    try {
        ListFacesRequest facesRequest = ListFacesRequest.builder()
            .collectionId(collectionId)
            .maxResults(10)
            .build();

        ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
        List<Face> faces = facesResponse.faces();
        for (Face face: faces) {
            System.out.println("Confidence level there is a face:
"+face.confidence());
            System.out.println("The face Id value is "+face.faceId());
        }
    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListFaces](#) in *AWS SDK for Java 2.x API Reference*.

## Recognize celebrities in an image

The following code example shows how to recognize celebrities in an image with Amazon Rekognition.

For more information, see [Recognizing celebrities in an image](#).

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void recognizeAllCelebrities(RekognitionClient rekClient, String sourceImage) {  
  
    try {  
        InputStream sourceStream = new FileInputStream(sourceImage);  
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);  
        Image souImage = Image.builder()  
            .bytes(sourceBytes)  
            .build();  
  
        RecognizeCelebritiesRequest request = RecognizeCelebritiesRequest.builder()  
            .image(souImage)  
            .build();  
  
        RecognizeCelebritiesResponse result =  
rekClient.recognizeCelebrities(request) ;  
        List<Celebrity> celebs=result.celebrityFaces();  
        System.out.println(celebs.size() + " celebrity(s) were recognized.\n");  
        for (Celebrity celebrity: celebs) {  
            System.out.println("Celebrity recognized: " + celebrity.name());  
            System.out.println("Celebrity ID: " + celebrity.id());  
  
            System.out.println("Further information (if available):");  
            for (String url: celebrity.urls()){  
                System.out.println(url);  
            }  
            System.out.println();  
        }  
        System.out.println(result.unrecognizedFaces().size() + " face(s) were  
unrecognized.");  
  
    } catch (RekognitionException | FileNotFoundException e) {  
        System.out.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [RecognizeCelebrities](#) in *AWS SDK for Java 2.x API Reference*.

## Search for faces in a collection

The following code example shows how to search for faces in an Amazon Rekognition collection that match another face from the collection.

For more information, see [Searching for a face \(face ID\)](#).

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void searchFaceInCollection(RekognitionClient rekClient, String collectionId, String sourceImage) {  
  
    try {  
        InputStream sourceStream = new FileInputStream(new File(sourceImage));  
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);  
        Image souImage = Image.builder()  
            .bytes(sourceBytes)
```

```
.build();

SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
    .image(souImage)
    .maxFaces(10)
    .faceMatchThreshold(70F)
    .collectionId(collectionId)
    .build();

SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest) ;
System.out.println("Faces matching in the collection");
List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
for (FaceMatch face: faceImageMatches) {
    System.out.println("The similarity level is " + face.similarity());
    System.out.println();
}

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [SearchFaces](#) in *AWS SDK for Java 2.x API Reference*.

### Search for faces in a collection compared to a reference image

The following code example shows how to search for faces in an Amazon Rekognition collection compared to a reference image.

For more information, see [Searching for a face \(image\)](#).

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void searchFacebyId(RekognitionClient rekClient, String collectionId,
String faceId) {

    try {
        SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
            .collectionId(collectionId)
            .faceId(faceId)
            .faceMatchThreshold(70F)
            .maxFaces(2)
            .build();

        SearchFacesResponse imageResponse =
rekClient.searchFaces(searchFacesRequest) ;
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face: faceImageMatches) {
            System.out.println("The similarity level is " + face.similarity());
            System.out.println();
        }

    } catch (RekognitionException e) {
```

```
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [SearchFacesByImage](#) in *AWS SDK for Java 2.x API Reference*.

## Scenarios

### Detect information in videos

The following code example shows how to:

- Start Amazon Rekognition jobs to detect elements like people, objects, and text in videos.
- Check job status until jobs finish.
- Output the list of elements detected by each job.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get celebrity results from a video located in an Amazon S3 bucket.

```
public static void StartCelebrityDetection(RekognitionClient rekClient,
                                            NotificationChannel channel,
                                            String bucket,
                                            String video){
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
            .jobTag("Celebrities")
            .notificationChannel(channel)
            .video(vidOb)
            .build();

        StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient.startCelebrityRecognition(recognitionRequest);
        startJobId = startCelebrityRecognitionResult.jobId();

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void GetCelebrityDetectionResults(RekognitionClient rekClient) {
    try {
        String paginationToken=null;
```

```

GetCelebrityRecognitionResponse recognitionResponse = null;
boolean finished = false;
String status;
int yy=0 ;

do{
    if (recognitionResponse !=null)
        paginationToken = recognitionResponse.nextToken();

    GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
    .jobId(startJobId)
    .nextToken(paginationToken)
    .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
    .maxResults(10)
    .build();

    // Wait until the job succeeds
    while (!finished) {
        recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
        status = recognitionResponse.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

    // Proceed when the job is done - otherwise VideoMetadata is null.
    VideoMetadata videoMetaData=recognitionResponse.videoMetadata();
    System.out.println("Format: " + videoMetaData.format());
    System.out.println("Codec: " + videoMetaData.codec());
    System.out.println("Duration: " + videoMetaData.durationMillis());
    System.out.println("FrameRate: " + videoMetaData.frameRate());
    System.out.println("Job");

    List<CelebrityRecognition> celebs= recognitionResponse.celebrities();
    for (CelebrityRecognition celeb: celebs) {
        long seconds=celeb.timestamp()/1000;
        System.out.print("Sec: " + seconds + " ");
        CelebrityDetail details=celeb.celebrity();
        System.out.println("Name: " + details.name());
        System.out.println("Id: " + details.id());
        System.out.println();
    }

} while (recognitionResponse.nextToken() != null);

} catch(RekognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}

```

Detect labels in a video by a label detection operation.

```

public static void startLabels(RekognitionClient rekClient,
                            NotificationChannel channel,

```

```
        String bucket,
        String video) {
    try {
        S3Object s3obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidOb)
            .minConfidence(50F)
            .build();

        StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

        boolean ans = true;
        String status = "";
        int yy = 0;
        while (ans) {

            GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .maxResults(10)
                .build();

            GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
            status = result.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                ans = false;
            else
                System.out.println(yy +" status is: "+status);

            Thread.sleep(1000);
            yy++;
        }

        System.out.println(startJobId +" status is: "+status);
    } catch(RekognitionException | InterruptedException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getLabelJob(RekognitionClient rekClient, SqSClient sqs, String
queueUrl) {

    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();
    }
```

```
if (!messages.isEmpty()) {
    for (Message message: messages) {
        String notification = message.body();

        // Get the status and job id from the notification
        ObjectMapper mapper = new ObjectMapper();
        JsonNode jsonMessageTree = mapper.readTree(notification);
        JsonNode messageBodyText = jsonMessageTree.get("Message");
        ObjectMapper operationResultMapper = new ObjectMapper();
        JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
        JsonNode operationJobId = jsonResultTree.get("JobId");
        JsonNode operationStatus = jsonResultTree.get("Status");
        System.out.println("Job found in JSON is " + operationJobId);

        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
    .queueUrl(queueUrl)
    .build();

        String jobId = operationJobId.textValue();
        if (startJobId.compareTo(jobId)==0) {
            System.out.println("Job id: " + operationJobId );
            System.out.println("Status : " + operationStatus.toString());

            if (operationStatus.asText().equals("SUCCEEDED"))
                GetResultsLabels(rekClient);
            else
                System.out.println("Video analysis failed");

            sqs.deleteMessage(deleteMessageRequest);
        }
        else{
            System.out.println("Job received was not job " + startJobId);
            sqs.deleteMessage(deleteMessageRequest);
        }
    }
}

} catch(RekognitionException e) {
    e.getMessage();
    System.exit(1);
} catch (JsonMappingException e) {
    e.printStackTrace();
} catch (JsonProcessingException e) {
    e.printStackTrace();
}
}

// Gets the job results by calling GetLabelDetection
private static void GetResultsLabels(RekognitionClient rekClient) {

    int maxResults=10;
    String paginationToken=null;
    GetLabelDetectionResponse labelDetectionResult=null;

    try {
        do {
            if (labelDetectionResult !=null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest=
GetLabelDetectionRequest.builder()
```

```

.jobId(startJobId)
.sortBy(LabelDetectionSortBy.TIMESTAMP)
.maxResults(maxResults)
.nextToken(paginationToken)
.build();

labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
VideoMetadata videoMetaData=labelDetectionResult.videoMetadata();
System.out.println("Format: " + videoMetaData.format());
System.out.println("Codec: " + videoMetaData.codec());
System.out.println("Duration: " + videoMetaData.durationMillis());
System.out.println("FrameRate: " + videoMetaData.frameRate());

List<LabelDetection> detectedLabels= labelDetectionResult.labels();
for (LabelDetection detectedLabel: detectedLabels) {
    long seconds=detectedLabel.timestamp();
    Label label=detectedLabel.label();
    System.out.println("Millisecond: " + seconds + " ");

    System.out.println("    Label:" + label.name());
    System.out.println("    Confidence:" +
detectedLabel.label().confidence().toString());

    List<Instance> instances = label.instances();
    System.out.println("    Instances of " + label.name());

    if (instances.isEmpty()) {
        System.out.println("        " + "None");
    } else {
        for (Instance instance : instances) {
            System.out.println("        Confidence: " +
instance.confidence().toString());
            System.out.println("        Bounding box: " +
instance.boundingBox().toString());
        }
    }
    System.out.println("    Parent labels for " + label.name() + ":");

    List<Parent> parents = label.parents();

    if (parents.isEmpty()) {
        System.out.println("        None");
    } else {
        for (Parent parent : parents) {
            System.out.println("        " + parent.name());
        }
    }
    System.out.println();
}
} while (labelDetectionResult !=null && labelDetectionResult.nextToken() !=

null);

} catch(RekognitionException e) {
    e.getMessage();
    System.exit(1);
}
}

```

Detect faces in a video stored in an Amazon S3 bucket.

```
public static void startLabels(RekognitionClient rekClient,
                            NotificationChannel channel,
                            String bucket,
                            String video) {
```

```
try {
    S3Object s3obj = S3Object.builder()
        .bucket(bucket)
        .name(video)
        .build();

    Video vid0b = Video.builder()
        .s3Object(s3obj)
        .build();

    StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
        .jobTag("DetectingLabels")
        .notificationChannel(channel)
        .video(vid0b)
        .minConfidence(50F)
        .build();

    StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
    startJobId = labelDetectionResponse.jobId();

    boolean ans = true;
    String status = "";
    int yy = 0;
    while (ans) {

        GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
            .jobId(startJobId)
            .maxResults(10)
            .build();

        GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
        status = result.jobStatusAsString();

        if (status.compareTo("SUCCEEDED") == 0)
            ans = false;
        else
            System.out.println(yy +" status is: "+status);

        Thread.sleep(1000);
        yy++;
    }

    System.out.println(startJobId +" status is: "+status);
} catch(RekognitionException | InterruptedException e) {
    e.getMessage();
    System.exit(1);
}
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs, String
queueUrl) {

    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();

    try {
        messages = sqs.receiveMessage(messageRequest).messages();
        if (!messages.isEmpty()) {
```

```

        for (Message message: messages) {
            String notification = message.body();

            // Get the status and job id from the notification
            ObjectMapper mapper = new ObjectMapper();
            JsonNode jsonMessageTree = mapper.readTree(notification);
            JsonNode messageBodyText = jsonMessageTree.get("Message");
            ObjectMapper operationResultMapper = new ObjectMapper();
            JsonNode jsonResultTree =
                operationResultMapper.readTree(messageBodyText.textValue());
            JsonNode operationJobId = jsonResultTree.get("JobId");
            JsonNode operationStatus = jsonResultTree.get("Status");
            System.out.println("Job found in JSON is " + operationJobId);

            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            String jobId = operationJobId.textValue();
            if (startJobId.compareTo(jobId)==0) {
                System.out.println("Job id: " + operationJobId );
                System.out.println("Status : " + operationStatus.toString());

                if (operationStatus.asText().equals("SUCCEEDED"))
                    GetResultsLabels(rekClient);
                else
                    System.out.println("Video analysis failed");

                sqs.deleteMessage(deleteMessageRequest);
            }

            else{
                System.out.println("Job received was not job " + startJobId);
                sqs.deleteMessage(deleteMessageRequest);
            }
        }

    } catch(RekognitionException e) {
        e.getMessage();
        System.exit(1);
    } catch (JsonMappingException e) {
        e.printStackTrace();
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }
}

// Gets the job results by calling GetLabelDetection
private static void GetResultsLabels(RekognitionClient rekClient) {

    int maxResults=10;
    String paginationToken=null;
    GetLabelDetectionResponse labelDetectionResult=null;

    try {
        do {
            if (labelDetectionResult !=null)
                paginationToken = labelDetectionResult.nextToken();

            GetLabelDetectionRequest labelDetectionRequest=
GetLabelDetectionRequest.builder()
                    .jobId(startJobId)
                    .sortBy(LabelDetectionSortBy.TIMESTAMP)

```

```

        .maxResults(maxResults)
        .nextToken(paginationToken)
        .build();

    labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
        VideoMetaData videoMetaData=labelDetectionResult.videoMetadata();
        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
        System.out.println("Duration: " + videoMetaData.durationMillis());
        System.out.println("FrameRate: " + videoMetaData.frameRate());

        List<LabelDetection> detectedLabels= labelDetectionResult.labels();
        for (LabelDetection detectedLabel: detectedLabels) {
            long seconds=detectedLabel.timestamp();
            Label label=detectedLabel.label();
            System.out.println("Millisecond: " + seconds + " ");

            System.out.println("    Label:" + label.name());
            System.out.println("    Confidence:" +
detectedLabel.label().confidence().toString());

            List<Instance> instances = label.instances();
            System.out.println("    Instances of " + label.name());

            if (instances.isEmpty()) {
                System.out.println("        " + "None");
            } else {
                for (Instance instance : instances) {
                    System.out.println("        Confidence: " +
instance.confidence().toString());
                    System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                }
            }
            System.out.println("    Parent labels for " + label.name() + ":");

            List<Parent> parents = label.parents();

            if (parents.isEmpty()) {
                System.out.println("        None");
            } else {
                for (Parent parent : parents) {
                    System.out.println("        " + parent.name());
                }
            }
            System.out.println();
        }
    } while (labelDetectionResult !=null && labelDetectionResult.nextToken() !=

null);

} catch(RekognitionException e) {
    e.getMessage();
    System.exit(1);
}
}

```

Detect inappropriate or offensive content in a video stored in an Amazon S3 bucket.

```
S3Object s3Obj = S3Object.builder()
    .bucket(bucket)
    .name(video)
    .build();

Video vid0b = Video.builder()
    .s3Object(s3Obj)
    .build();

StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
    .jobTag("Moderation")
    .notificationChannel(channel)
    .video(vid0b)
    .build();

StartContentModerationResponse startModDetectionResult =
rekClient.startContentModeration(modDetectionRequest);
startJobId=startModDetectionResult.jobId();

} catch(RekognitionException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}

public static void GetModResults(RekognitionClient rekClient) {

try {
    String paginationToken=null;
    GetContentModerationResponse modDetectionResponse=null;
    boolean finished = false;
    String status;
    int yy=0 ;

    do{
        if (modDetectionResponse !=null)
            paginationToken = modDetectionResponse.nextToken();

        GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
    .jobId(startJobId)
    .nextToken(paginationToken)
    .maxResults(10)
    .build();

        // Wait until the job succeeds
        while (!finished) {
            modDetectionResponse = rekClient.getContentModeration(modRequest);
            status = modDetectionResponse.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(yy + " status is: " + status);
                Thread.sleep(1000);
            }
            yy++;
        }

        finished = false;

        // Proceed when the job is done - otherwise VideoMetadata is null
        VideoMetadata videoMetaData=modDetectionResponse.videoMetadata();
        System.out.println("Format: " + videoMetaData.format());
        System.out.println("Codec: " + videoMetaData.codec());
    }
}
```

```
System.out.println("Duration: " + videoMetaData.durationMillis());
System.out.println("FrameRate: " + videoMetaData.frameRate());
System.out.println("Job");

List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
for (ContentModerationDetection mod: mods) {
    long seconds=mod.timestamp()/1000;
    System.out.print("Mod label: " + seconds + " ");
    System.out.println(mod.moderationLabel().toString());
    System.out.println();
}

} while (modDetectionResponse !=null && modDetectionResponse.nextToken() !=null);

} catch(RekognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

Detect technical cue segments and shot detection segments in a video stored in an Amazon S3 bucket.

```
public static void StartSegmentDetection (RekognitionClient rekClient,
                                         NotificationChannel channel,
                                         String bucket,
                                         String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartShotDetectionFilter cueDetectionFilter =
StartShotDetectionFilter.builder()
            .minSegmentConfidence(60F)
            .build();

        StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
StartTechnicalCueDetectionFilter.builder()
            .minSegmentConfidence(60F)
            .build();

        StartSegmentDetectionFilters filters =
StartSegmentDetectionFilters.builder()
            .shotFilter(cueDetectionFilter)
            .technicalCueFilter(technicalCueDetectionFilter)
            .build();

        StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .segmentTypes(SegmentType.TECHNICAL_CUE , SegmentType.SHOT)
            .video(vidOb)
            .filters(filters)
            .build();
    }
}
```

```
        StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
        startJobId = segDetectionResponse.jobId();

    } catch(RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getSegmentResults(RekognitionClient rekClient) {

    try {
        String paginationToken = null;
        GetSegmentDetectionResponse segDetectionResponse = null;
        boolean finished = false;
        String status;
        int yy = 0;

        do {
            if (segDetectionResponse != null)
                paginationToken = segDetectionResponse.nextToken();

            GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
                status = segDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }
            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            List<VideoMetadata> videoMetaDataTable =
segDetectionResponse.videoMetadata();
            for (VideoMetadata metaData : videoMetaDataTable) {
                System.out.println("Format: " + metaData.format());
                System.out.println("Codec: " + metaData.codec());
                System.out.println("Duration: " + metaData.durationMillis());
                System.out.println("FrameRate: " + metaData.frameRate());
                System.out.println("Job");
            }
        }

        List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
        for (SegmentDetection detectedSegment : detectedSegments) {
            String type = detectedSegment.type().toString();
            if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
                System.out.println("Technical Cue");
                TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
            }
        }
    }
}
```

```
        System.out.println("\tType: " + segmentCue.type());
        System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
    }

    if (type.contains(SegmentType.SHOT.toString())) {
        System.out.println("Shot");
        ShotSegment segmentShot = detectedSegment.shotSegment();
        System.out.println("\tIndex " + segmentShot.index());
        System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
    }

    long seconds = detectedSegment.durationMillis();
    System.out.println("\tDuration : " + seconds + " milliseconds");
    System.out.println("\tStart time code: " +
detectedSegment.startTimecodeSMPTE());
    System.out.println("\tEnd time code: " +
detectedSegment.endTimecodeSMPTE());
    System.out.println("\tDuration time code: " +
detectedSegment.durationSMPTE());
    System.out.println();
}

} while (segDetectionResponse !=null && segDetectionResponse.nextToken() != null);

} catch(RekognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

Detect text in a video stored in a video stored in an Amazon S3 bucket.

```
public static void startTextLabels(RekognitionClient rekClient,
                                    NotificationChannel channel,
                                    String bucket,
                                    String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidOb)
            .build();

        StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}

public static void GetTextResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetTextDetectionResponse textDetectionResponse=null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (textDetectionResponse !=null)
                paginationToken = textDetectionResponse.nextToken();

            GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                    .jobId(startJobId)
                    .nextToken(paginationToken)
                    .maxResults(10)
                    .build();

            // Wait until the job succeeds.
            while (!finished) {
                textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                status = textDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData=textDetectionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());
            System.out.println("Job");

            List<TextDetectionResult> labels=
textDetectionResponse.textDetections();
            for (TextDetectionResult detectedText: labels) {
                System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
                System.out.println("Id : " + detectedText.textDetection().id());
                System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
                System.out.println("Type: " + detectedText.textDetection().type());
                System.out.println("Text: " +
detectedText.textDetection().detectedText());
                System.out.println();
            }

        } while (textDetectionResponse !=null &&
textDetectionResponse.nextToken() != null);

    } catch(RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

Detect people in a video stored in a video stored in an Amazon S3 bucket.

```
public static void startPersonLabels(RekognitionClient rekClient,
                                     NotificationChannel channel,
                                     String bucket,
                                     String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
            .jobTag("DetectingLabels")
            .video(vidOb)
            .notificationChannel(channel)
            .build();

        StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
        startJobId = labelDetectionResponse.jobId();

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void GetPersonDetectionResults(RekognitionClient rekClient) {
    try {
        String paginationToken=null;
        GetPersonTrackingResponse personTrackingResult=null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (personTrackingResult !=null)
                paginationToken = personTrackingResult.nextToken();

            GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
            .jobId(startJobId)
            .nextToken(paginationToken)
            .maxResults(10)
            .build();

            // Wait until the job succeeds
            while (!finished) {

                personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                status = personTrackingResult.jobStatusAsString();
            }
        }
    }
}
```

```
        if (status.compareTo("SUCCEEDED") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + status);
            Thread.sleep(1000);
        }
        yy++;
    }

    finished = false;

// Proceed when the job is done - otherwise VideoMetadata is null
VideoMetadata videoMetaData = personTrackingResult.videoMetadata();

System.out.println("Format: " + videoMetaData.format());
System.out.println("Codec: " + videoMetaData.codec());
System.out.println("Duration: " + videoMetaData.durationMillis());
System.out.println("FrameRate: " + videoMetaData.frameRate());
System.out.println("Job");

List<PersonDetection> detectedPersons= personTrackingResult.persons();
for (PersonDetection detectedPerson: detectedPersons) {

    long seconds=detectedPerson.timestamp()/1000;
    System.out.print("Sec: " + seconds + " ");
    System.out.println("Person Identifier: " +
detectedPerson.person().index());
    System.out.println();
}

} while (personTrackingResult !=null && personTrackingResult.nextToken() !=null);

} catch(RekognitionException | InterruptedException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [GetCelebrityRecognition](#)
  - [GetContentModeration](#)
  - [GetLabelDetection](#)
  - [GetPersonTracking](#)
  - [GetSegmentDetection](#)
  - [GetTextDetection](#)
  - [StartCelebrityRecognition](#)
  - [StartContentModeration](#)
  - [StartLabelDetection](#)
  - [StartPersonTracking](#)
  - [StartSegmentDetection](#)
  - [StartTextDetection](#)

## Amazon S3 examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Simple Storage Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3406\)](#)
- [Scenarios \(p. 3425\)](#)

## Actions

### Add CORS rules to a bucket

The following code example shows how to add cross-origin resource sharing (CORS) rules to an S3 bucket.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
    try {
        DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
        .bucket(bucketName)
        .expectedBucketOwner(accountId)
        .build();

        s3.deleteBucketCors(bucketCorsRequest) ;

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getBucketCorsInformation(S3Client s3, String bucketName, String
accountId) {
    try {
        GetBucketCorsRequest bucketCorsRequest = GetBucketCorsRequest.builder()
        .bucket(bucketName)
        .expectedBucketOwner(accountId)
        .build();

        GetBucketCorsResponse corsResponse = s3.getBucketCors(bucketCorsRequest);
        List<CORSRule> corsRules = corsResponse.corsRules();
        for (CORSRule rule: corsRules) {
            System.out.println("allowOrigins: "+rule.allowedOrigins());
            System.out.println("AllowedMethod: "+rule.allowedMethods());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
public static void setCorsInformation(S3Client s3, String bucketName, String accountId) {

    List<String> allowMethods = new ArrayList<>();
    allowMethods.add("PUT");
    allowMethods.add("POST");
    allowMethods.add("DELETE");

    List<String> allowOrigins = new ArrayList<>();
    allowOrigins.add("http://example.com");
    try {
        // Define CORS rules.
        CORSRule corsRule = CORSRule.builder()
            .allowedMethods(allowMethods)
            .allowedOrigins(allowOrigins)
            .build();

        List<CORSRule> corsRules = new ArrayList<>();
        corsRules.add(corsRule);
        CORSConfiguration configuration = CORSConfiguration.builder()
            .corsRules(corsRules)
            .build();

        PutBucketCorsRequest putBucketCorsRequest = PutBucketCorsRequest.builder()
            .bucket(bucketName)
            .corsConfiguration(configuration)
            .expectedBucketOwner(accountId)
            .build();

        s3.putBucketCors(putBucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutBucketCors](#) in *AWS SDK for Java 2.x API Reference*.

## Add a lifecycle configuration to a bucket

The following code example shows how to add a lifecycle configuration to an S3 bucket.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void setLifecycleConfig(S3Client s3, String bucketName, String accountId) {

    try {
        // Create a rule to archive objects with the "glacierobjects/" prefix to
        // Amazon S3 Glacier.
        LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
            .prefix("glacierobjects/")
            .build();

        Transition transition = Transition.builder()
            .storageClass(TransitionStorageClass.GLACIER)
            .days(0)
```

```
.build();

LifecycleRule rule1 = LifecycleRule.builder()
    .id("Archive immediately rule")
    .filter(ruleFilter)
    .transitions(transition)
    .status(ExpirationStatus.ENABLED)
    .build();

// Create a second rule.
Transition transition2 = Transition.builder()
    .storageClass(TransitionStorageClass.GLACIER)
    .days(0)
    .build();

List<Transition> transitionList = new ArrayList<>();
transitionList.add(transition2);

LifecycleRuleFilter ruleFilter2 = LifecycleRuleFilter.builder()
    .prefix("glacierobjects/")
    .build();

LifecycleRule rule2 = LifecycleRule.builder()
    .id("Archive and then delete rule")
    .filter(ruleFilter2)
    .transitions(transitionList)
    .status(ExpirationStatus.ENABLED)
    .build();

// Add the LifecycleRule objects to an ArrayList.
ArrayList<LifecycleRule> ruleList = new ArrayList<>();
ruleList.add(rule1);
ruleList.add(rule2);

BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
    .rules(ruleList)
    .build();

PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest =
PutBucketLifecycleConfigurationRequest.builder()
    .bucket(bucketName)
    .lifecycleConfiguration(lifecycleConfiguration)
    .expectedBucketOwner(accountId)
    .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

// Retrieve the configuration and add a new rule.
public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId){

    try {
        GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest =
GetBucketLifecycleConfigurationRequest.builder()
        .bucket(bucketName)
        .expectedBucketOwner(accountId)
        .build();
    }
}
```

```
        GetBucketLifecycleConfigurationResponse response =
s3.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
        List<LifecycleRule> newList = new ArrayList<>();
        List<LifecycleRule> rules = response.rules();
        for (LifecycleRule rule: rules) {
            newList.add(rule);
        }

        // Add a new rule with both a prefix predicate and a tag predicate.
        LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
            .prefix("YearlyDocuments/")
            .build();

        Transition transition = Transition.builder()
            .storageClass(TransitionStorageClass.GLACIER)
            .days(3650)
            .build();

        LifecycleRule rule1 = LifecycleRule.builder()
            .id("NewRule")
            .filter(ruleFilter)
            .transitions(transition)
            .status(ExpirationStatus.ENABLED)
            .build();

        // Add the new rule to the list.
        newList.add(rule1);
        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
            .rules(newList)
            .build();

        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest =
PutBucketLifecycleConfigurationRequest.builder()
            .bucket(bucketName)
            .lifecycleConfiguration(lifecycleConfiguration)
            .expectedBucketOwner(accountId)
            .build();

        s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// Delete the configuration from the Amazon S3 bucket.
public static void deleteLifecycleConfig(S3Client s3, String bucketName, String
accountId) {

    try {
        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest.builder()
            .bucket(bucketName)
            .expectedBucketOwner(accountId)
            .build();

        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- For API details, see [PutBucketLifecycleConfiguration](#) in *AWS SDK for Java 2.x API Reference*.

## Add a policy to a bucket

The following code example shows how to add a policy to an S3 bucket.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void setPolicy(S3Client s3, String bucketName, String policyText) {  
  
    System.out.println("Setting policy:");  
    System.out.println("----");  
    System.out.println(policyText);  
    System.out.println("----");  
    System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);  
  
    try {  
        PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()  
            .bucket(bucketName)  
            .policy(policyText)  
            .build();  
  
        s3.putBucketPolicy(policyReq);  
  
    } catch (S3Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
  
    System.out.println("Done!");  
}  
  
// Loads a JSON-formatted policy from a file  
public static String getBucketPolicyFromFile(String policyFile) {  
  
    StringBuilder fileText = new StringBuilder();  
    try {  
        List<String> lines = Files.readAllLines(Paths.get(policyFile),  
StandardCharsets.UTF_8);  
        for (String line : lines) {  
            fileText.append(line);  
        }  
  
    } catch (IOException e) {  
        System.out.format("Problem reading file: \"%s\"", policyFile);  
        System.out.println(e.getMessage());  
    }  
  
    try {  
        final JsonParser parser = new  
ObjectMapper().getFactory().createParser(fileText.toString());  
        while (parser.nextToken() != null) {  
        }  
  
    } catch (IOException jpe) {  
        jpe.printStackTrace();  
    }  
}
```

```
        }
    }  
    return fileText.toString();  
}
```

- For API details, see [PutBucketPolicy in AWS SDK for Java 2.x API Reference](#).

## Copy an object from one bucket to another

The following code example shows how to copy an S3 object from one bucket to another.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String copyBucketObject (S3Client s3, String fromBucket, String  
objectKey, String toBucket) {  
  
    String encodedUrl = "";  
    try {  
        encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,  
StandardCharsets.UTF_8.toString());  
  
    } catch (UnsupportedEncodingException e) {  
        System.out.println("URL could not be encoded: " + e.getMessage());  
    }  
  
    CopyObjectRequest copyReq = CopyObjectRequest.builder()  
        .copySourceIfMatch(encodedUrl)  
        .destinationBucket(toBucket)  
        .destinationKey(objectKey)  
        .build();  
  
    try {  
        CopyObjectResponse copyRes = s3.copyObject(copyReq);  
        return copyRes.copyObjectResult().toString();  
  
    } catch (S3Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [CopyObject in AWS SDK for Java 2.x API Reference](#).

## Create a bucket

The following code example shows how to create an S3 bucket.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createBucket( S3Client s3Client, String bucketName) {
```

```
try {
    S3Waiter s3Waiter = s3Client.waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    s3Client.createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    WaiterResponse<HeadBucketResponse> waiterResponse =
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println(bucketName + " is ready");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a policy from a bucket

The following code example shows how to delete a policy from an S3 bucket.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Delete the bucket policy.
public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {

    DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketPolicy(delReq);
        System.out.println("Done!");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteBucketPolicy](#) in *AWS SDK for Java 2.x API Reference*.

## Delete an empty bucket

The following code example shows how to delete an empty S3 bucket.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

- For API details, see [DeleteBucket](#) in *AWS SDK for Java 2.x API Reference*.

## Delete multiple objects

The following code example shows how to delete multiple objects from an S3 bucket.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteBucketObjects(S3Client s3, String bucketName) {

    // Upload three sample objects to the specified Amazon S3 bucket.
    ArrayList<ObjectIdentifier> keys = new ArrayList<>();
    PutObjectRequest putOb;
    ObjectIdentifier objectId;

    for (int i = 0; i < 3; i++) {
        String keyName = "delete object example " + i;
        objectId = ObjectIdentifier.builder()
            .key(keyName)
            .build();

        putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        s3.putObject(putOb, RequestBody.fromString(keyName));
        keys.add(objectId);
    }

    System.out.println(keys.size() + " objects successfully created.");

    // Delete multiple objects in one request.
    Delete del = Delete.builder()
        .objects(keys)
        .build();

    try {
        DeleteObjectsRequest multiObjectDeleteRequest =
DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(del)
            .build();
    }
}
```

```
s3.deleteObjects(multiObjectDeleteRequest);
System.out.println("Multiple objects are deleted!");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

- For API details, see [DeleteObjects](#) in *AWS SDK for Java 2.x API Reference*.

## Delete the website configuration from a bucket

The following code example shows how to delete the website configuration from an S3 bucket.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName) {

    DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketWebsite(delReq);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.out.println("Failed to delete website configuration!");
        System.exit(1);
    }
}
```

- For API details, see [DeleteBucketWebsite](#) in *AWS SDK for Java 2.x API Reference*.

## Determine the existence and content type of an object

The following code example shows how to determine the existence and content type of an object in an S3 bucket.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getContentType (S3Client s3, String bucketName, String keyName)
{
    try {
        HeadObjectRequest objectRequest = HeadObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
```

```
.build();

HeadObjectResponse objectHead = s3.headObject(objectRequest);
String type = objectHead.contentType();
System.out.println("The object content type is "+type);

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [HeadObject](#) in *AWS SDK for Java 2.x API Reference*.

## Get an object from a bucket

The following code example shows how to read data from an object in an S3 bucket.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Read data as a byte array.

```
public static void getObjectBytes (S3Client s3, String bucketName, String keyName,
String path) {

    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
        byte[] data = objectBytes.asByteArray();

        // Write the data to a local file.
        File myFile = new File(path );
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
        os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Read tags that belong to an object.

```
public static void listTags(S3Client s3, String bucketName, String keyName) {

    try {
        GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
```

```
.builder()  
.key(keyName)  
.bucket(bucketName)  
.build();  
  
GetObjectTaggingResponse tags = s3.getObjectTagging(getTaggingRequest);  
List<Tag> tagSet= tags.tagSet();  
for (Tag tag : tagSet) {  
    System.out.println(tag.key());  
    System.out.println(tag.value());  
}  
  
} catch (S3Exception e) {  
    System.err.println(e.awsErrorDetails().errorMessage());  
    System.exit(1);  
}  
}
```

#### Get a URL for an object.

```
public static void getURL(S3Client s3, String bucketName, String keyName ) {  
  
try {  
    GetUrlRequest request = GetUrlRequest.builder()  
        .bucket(bucketName)  
        .key(keyName)  
        .build();  
  
    URL url = s3.utilities().getUrl(request);  
    System.out.println("The URL for "+keyName +" is "+ url);  
  
} catch (S3Exception e) {  
    System.err.println(e.awsErrorDetails().errorMessage());  
    System.exit(1);  
}  
}
```

#### Get an object by using the S3Presigner client object.

```
public static void getPresignedUrl(S3Presigner presigner, String bucketName,  
String keyName ) {  
  
try {  
    GetObjectRequest getObjectRequest = GetObjectRequest.builder()  
        .bucket(bucketName)  
        .key(keyName)  
        .build();  
  
    GetObjectPresignRequest getObjectPresignRequest =  
GetObjectPresignRequest.builder()  
        .signatureDuration(Duration.ofMinutes(60))  
        .getObjectRequest(getObjectRequest)  
        .build();  
  
    PresignedGetObjectRequest presignedGetObjectRequest =  
presigner.presignGetObject(getObjectPresignRequest);  
    String theUrl = presignedGetObjectRequest.url().toString();  
    System.out.println("Presigned URL: " + theUrl);  
    HttpURLConnection connection = (HttpURLConnection)  
presignedGetObjectRequest.url().openConnection();  
    presignedGetObjectRequest.httpRequest().headers().forEach((header,  
values) -> {
```

```
        values.forEach(value -> {
            connection.addRequestProperty(header, value);
        });
    });

    // Send any request payload that the service needs (not needed when
    isBrowserExecutable is true).
    if (presignedGetObjectRequest.signedPayload().isPresent()) {
        connection.setDoOutput(true);

        try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
            OutputStream httpOutputStream = connection.getOutputStream()) {
            IoUtils.copy(signedPayload, httpOutputStream);
        }
    }

    // Download the result of executing the request.
    try (InputStream content = connection.getInputStream()) {
        System.out.println("Service returned response: ");
        IoUtils.copy(content, System.out);
    }

} catch (S3Exception | IOException e) {
    e.printStackTrace();
}
}
```

- For API details, see [GetObject](#) in *AWS SDK for Java 2.x API Reference*.

## Get the ACL of a bucket

The following code example shows how to get the access control list (ACL) of an S3 bucket.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String getBucketACL(S3Client s3, String objectKey, String bucketName) {
    try {
        GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
        List<Grant> grants = aclRes.grants();
        String grantee = "";
        for (Grant grant : grants) {
            System.out.format(" %s: %s\n", grant.grantee().id(),
grant.permission());
            grantee = grant.grantee().id();
        }

        return grantee;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        }

    return "";
}
```

- For API details, see [GetBucketAcl](#) in *AWS SDK for Java 2.x API Reference*.

## Get the policy for a bucket

The following code example shows how to get the policy for an S3 bucket.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String getPolicy(S3Client s3, String bucketName) {

    String policyText;
    System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
    GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
        policyText = policyRes.policy();
        return policyText;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- For API details, see [GetBucketPolicy](#) in *AWS SDK for Java 2.x API Reference*.

## List in-progress multipart uploads

The following code example shows how to list in-progress multipart uploads to an S3 bucket.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listUploads( S3Client s3, String bucketName) {

    try {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();
```

```
        ListMulticastUploadsResponse response =
s3.listMulticastUploads(listMulticastUploadsRequest);
        List<MulticastUpload> uploads = response.uploads();
        for (MulticastUpload upload: uploads) {
            System.out.println("Upload in progress: Key = \\" + upload.key() + "\","
id = " + upload.uploadId());
        }

    } catch (S3Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- For API details, see [ListMulticastUploads](#) in *AWS SDK for Java 2.x API Reference*.

## List objects in a bucket

The following code example shows how to list objects in an S3 bucket.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listBucketObjects(S3Client s3, String bucketName ) {

    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.print("\n The name of the key is " + myValue.key());
            System.out.print("\n The object is " + calKb(myValue.size()) + " KBs");
            System.out.print("\n The owner is " + myValue.owner());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

//convert bytes to kbs.
private static long calKb(Long val) {
    return val/1024;
}
```

- For API details, see [ListObjects](#) in *AWS SDK for Java 2.x API Reference*.

## Restore an archived copy of an object

The following code example shows how to restore an archived copy of an object back into an S3 bucket.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void restoreS3Object(S3Client s3, String bucketName, String keyName,
String expectedBucketOwner) {

    try {
        RestoreRequest restoreRequest = RestoreRequest.builder()
            .days(10)

        .glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
            .build();

        RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
            .expectedBucketOwner(expectedBucketOwner)
            .bucket(bucketName)
            .key(keyName)
            .restoreRequest(restoreRequest)
            .build();

        s3.restoreObject(objectRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [RestoreObject](#) in *AWS SDK for Java 2.x API Reference*.

## Set a new ACL for a bucket

The following code example shows how to set a new access control list (ACL) for an S3 bucket.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void setBucketAcl(S3Client s3, String bucketName, String id) {

    try {
        Grant ownerGrant = Grant.builder()
            .grantee(builder -> builder.id(id))
            .type(Type.CANONICAL_USER)
            .permission(Permission.FULL_CONTROL)
            .build();

        List<Grant> grantList2 = new ArrayList<>();
        grantList2.add(ownerGrant);

        AccessControlPolicy acl = AccessControlPolicy.builder()
            .owner(builder -> builder.id(id))
            .grants(grantList2)
            .build();

        PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
```

```
.bucket(bucketName)
.accessControlPolicy(acl)
.build();

s3.putBucketAcl(putAclReq);

} catch (S3Exception e) {
    e.printStackTrace();
    System.exit(1);
}
}
```

- For API details, see [PutBucketAcl](#) in *AWS SDK for Java 2.x API Reference*.

## Set the website configuration for a bucket

The following code example shows how to set the website configuration for an S3 bucket.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void setWebsiteConfig( S3Client s3, String bucketName, String
indexDoc) {

    try {
        WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
            .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
            .build();

        PutBucketWebsiteRequest pubWebsiteReq = PutBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .websiteConfiguration(websiteConfig)
            .build();

        s3.putBucketWebsite(pubWebsiteReq);
        System.out.println("The call was successful");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutBucketWebsite](#) in *AWS SDK for Java 2.x API Reference*.

## Upload an object to a bucket

The following code example shows how to upload an object to an S3 bucket.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Upload an object to a bucket.

```
public static String putS3Object(S3Client s3, String bucketName, String objectKey,  
String objectPath) {  
  
    try {  
        Map<String, String> metadata = new HashMap<>();  
        metadata.put("x-amz-meta-myVal", "test");  
        PutObjectRequest putOb = PutObjectRequest.builder()  
            .bucket(bucketName)  
            .key(objectKey)  
            .metadata(metadata)  
            .build();  
  
        PutObjectResponse response = s3.putObject(putOb,  
RequestBody.fromBytes(getObjectFile(objectPath)));  
        return response.eTag();  
  
    } catch (S3Exception e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
  
    return "";  
}  
  
// Return a byte array.  
private static byte[] getObjectFile(String filePath) {  
  
    FileInputStream fileInputStream = null;  
    byte[] bytesArray = null;  
  
    try {  
        File file = new File(filePath);  
        bytesArray = new byte[(int) file.length()];  
        fileInputStream = new FileInputStream(file);  
        fileInputStream.read(bytesArray);  
  
    } catch (IOException e) {  
        e.printStackTrace();  
    } finally {  
        if (fileInputStream != null) {  
            try {  
                fileInputStream.close();  
            } catch (IOException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
  
    return bytesArray;  
}
```

Upload an object to a bucket and set tags.

```
public static void putS3ObjectTags(S3Client s3, String bucketName, String  
objectKey, String objectPath) {  
  
    try {  
  
        Tag tag1 = Tag.builder()  
            .key("Tag 1")  
            .value("This is tag 1")  
            .build();  
  
    }
```

```
Tag tag2 = Tag.builder()
    .key("Tag 2")
    .value("This is tag 2")
    .build();

List<Tag> tags = new ArrayList<>();
tags.add(tag1);
tags.add(tag2);

Tagging allTags = Tagging.builder()
    .tagSet(tags)
    .build();

PutObjectRequest putOb = PutObjectRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .tagging(allTags)
    .build();

s3.putObject(putOb, RequestBody.fromBytes(getObjectFile(objectPath)));

} catch (S3Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {

    try {
        GetObjectTaggingRequest taggingRequest = GetObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
        List<Tag> obTags = getTaggingRes.tagSet();
        for (Tag sinTag: obTags) {
            System.out.println("The tag key is: "+sinTag.key());
            System.out.println("The tag value is: "+sinTag.value());
        }

        // Replace the object's tags with two new tags.
        Tag tag3 = Tag.builder()
            .key("Tag 3")
            .value("This is tag 3")
            .build();

        Tag tag4 = Tag.builder()
            .key("Tag 4")
            .value("This is tag 4")
            .build();

        List<Tag> tags = new ArrayList<>();
        tags.add(tag3);
        tags.add(tag4);

        Tagging updatedTags = Tagging.builder()
            .tagSet(tags)
            .build();

        PutObjectTaggingRequest taggingRequest1 = PutObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
```

```
        .tagging(updatedTags)
        .build();

        s3.putObjectTagging(taggingRequest1);
        GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
        List<Tag> modTags = getTaggingRes2.tagSet();
        for (Tag sinTag: modTags) {
            System.out.println("The tag key is: "+sinTag.key());
            System.out.println("The tag value is: "+sinTag.value());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Upload an object to a bucket and set metadata.

```
public static String putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {

    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("author", "Mary Doe");
        metadata.put("version", "1.0.0.0");

        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .metadata(metadata)
            .build();

        PutObjectResponse response = s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));
        return response.eTag();

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    return "";
}

// Return a byte array.
private static byte[] getObjectFile(String filePath) {

    FileInputStream fileInputStream = null;
    byte[] bytesArray = null;

    try {
        File file = new File(filePath);
        bytesArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(bytesArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();

```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    return byteArray;
}
```

Upload an object to a bucket and set an object retention value.

```
public static void setRetentionPeriod(S3Client s3, String key, String bucket) {

    try{
        LocalDate localDate = LocalDate.parse("2020-07-17");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        ObjectLockRetention lockRetention = ObjectLockRetention.builder()
            .mode("COMPLIANCE")
            .retainUntilDate(instant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
    .bucket(bucket)
    .key(key)
    .bypassGovernanceRetention(true)
    .retention(lockRetention)
    .build();

        // To set Retention on an object, the Amazon S3 bucket must support object
        // locking, otherwise an exception is thrown.
        s3.putObjectRetention(retentionRequest);
        System.out.print("An object retention configuration was successfully placed
on the object");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [PutObject](#) in *AWS SDK for Java 2.x API Reference*.

## Scenarios

### Create a presigned URL

The following code example shows how to create a presigned URL for S3 and upload an object.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void signBucket(S3Presigner presigner, String bucketName, String
keyName) {
```

```
try {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(keyName)
        .contentType("text/plain")
        .build();

    PutObjectPresignRequest presignRequest = PutObjectPresignRequest.builder()
        .signatureDuration(Duration.ofMinutes(10))
        .putObjectRequest(objectRequest)
        .build();

    PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
    String myURL = presignedRequest.url().toString();
    System.out.println("Presigned URL to upload a file to: " +myURL);
    System.out.println("Which HTTP method needs to be used when uploading a
file: " + presignedRequest.httpRequest().method());

    // Upload content to the Amazon S3 bucket by using this URL.
    URL url = presignedRequest.url();

    // Create the connection and use it to upload the new object by using the
presigned URL.
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    connection.setDoOutput(true);
    connection.setRequestProperty("Content-Type", "text/plain");
    connection.setRequestMethod("PUT");
    OutputStreamWriter out = new
OutputStreamWriter(connection.getOutputStream());
    out.write("This text was uploaded as an object by using a presigned URL.");
    out.close();

    connection.getResponseCode();
    System.out.println("HTTP response code is " +
connection.getResponseCode());

} catch (S3Exception | IOException e) {
    e.printStackTrace();
}
}
```

## Get started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.
- Delete a bucket.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
 * Before running this Java V2 code example, set up your development environment,  
 including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * This Java code example performs the following tasks:  
 *  
 * 1. Creates an Amazon S3 bucket.  
 * 2. Uploads an object to the bucket.  
 * 3. Downloads the object to another local file.  
 * 4. Uploads an object using multipart upload.  
 * 5. List all objects located in the Amazon S3 bucket.  
 * 6. Copies the object to another Amazon S3 bucket.  
 * 7. Deletes the object from the Amazon S3 bucket.  
 * 8. Deletes the Amazon S3 bucket.  
 */  
  
public class S3Scenario {  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    public static void main(String[] args) throws IOException {  
  
        final String usage = "\n" +  
            "Usage:\n" +  
            "    <bucketName> <key> <objectPath> <savePath> <toBucket>\n\n" +  
            "Where:\n" +  
            "    <bucketName> - The Amazon S3 bucket to create.\n\n" +  
            "    <key> - The key to use.\n\n" +  
            "    <objectPath> - The path where the file is located (for example, C:/AWS/  
book2.pdf). "+  
            "    <savePath> - The path where the file is saved after it's downloaded (for  
example, C:/AWS/book2.pdf). " +  
            "    <toBucket> - An Amazon S3 bucket to where an object is copied to (for  
example, C:/AWS/book2.pdf). "  
  
        if (args.length != 5) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String bucketName = args[0];  
        String key = args[1];  
        String objectPath = args[2];  
        String savePath = args[3];  
        String toBucket = args[4] ;  
  
        ProfileCredentialsProvider credentialsProvider =  
ProfileCredentialsProvider.create();  
        Region region = Region.US_EAST_1;  
        S3Client s3 = S3Client.builder()  
            .region(region)  
            .credentialsProvider(credentialsProvider)  
            .build();  
  
        System.out.println(DASHES);  
        System.out.println("Welcome to the Amazon S3 example scenario.");  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("1. Create an Amazon S3 bucket.");  
        createBucket(s3, bucketName);  
        System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("2. Update a local file to the Amazon S3 bucket.");
uploadLocalFile(s3, bucketName, key, objectPath);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Download the object to another local file.");
getBytes (s3, bucketName, key, savePath);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Perform a multipart upload.");
String multipartKey = "multiPartKey";
multipartUpload(s3, toBucket, multipartKey);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. List all objects located in the Amazon S3 bucket.");
listAllObjects(s3, bucketName);
anotherListExample(s3, bucketName) ;
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Copy the object to another Amazon S3 bucket.");
copyBucketObject (s3, bucketName, key, toBucket);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the object from the Amazon S3 bucket.");
deleteObjectFromBucket(s3, bucketName, key);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Delete the Amazon S3 bucket.");
deleteBucket(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("All Amazon S3 operations were successfully performed");
System.out.println(DASHES);
s3.close();
}

// Create a bucket by using a S3Waiter object
public static void createBucket( S3Client s3Client, String bucketName) {
try {
    S3Waiter s3Waiter = s3Client.waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    s3Client.createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println(bucketName +" is ready");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

```
        }

    public static void deleteBucket(S3Client client, String bucket) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
            .bucket(bucket)
            .build();

        client.deleteBucket(deleteBucketRequest);
        System.out.println(bucket + " was deleted.");
    }

    /**
     * Upload an object in parts
     */
    private static void multipartUpload(S3Client s3, String bucketName, String key) {
        int mB = 1024 * 1024;
        // First create a multipart upload and get the upload id
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        System.out.println(uploadId);

        // Upload all the different parts of the object
        UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .partNumber(1).build();

        String etag1 = s3.uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB))).eTag();
        CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
            .uploadId(uploadId)
            .partNumber(2).build();
        String etag2 = s3.uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB))).eTag();
        CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

        // Call completeMultipartUpload operation to tell S3 to merge all uploaded
        // parts and finish the multipart operation.
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
            .parts(part1, part2)
            .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .multipartUpload(completedMultipartUpload)
            .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
```

```
}

private static ByteBuffer getRandomByteBuffer(int size) {
    byte[] b = new byte[size];
    new Random().nextBytes(b);
    return ByteBuffer.wrap(b);
}

// Return a byte array
private static byte[] getObjectFile(String filePath) {
    FileInputStream fileInputStream = null;
    byte[] bytesArray = null;

    try {
        File file = new File(filePath);
        bytesArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(bytesArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    return bytesArray;
}

public static void getgetBytes (S3Client s3, String bucketName, String keyName,
String path ) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
        byte[] data = objectBytes.asByteArray();

        // Write the data to a local file.
        File myFile = new File(path);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
        os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void uploadLocalFile(S3Client s3, String bucketName, String key,
String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
```

```
.key(key)
.build();

s3.putObject(objectRequest, RequestBody.fromBytes(getObjectFile(objectPath)));
}

public static void listAllObjects(S3Client s3, String bucketName) {

    ListObjectsV2Request listObjectsReqManual = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    boolean done = false;
    while (!done) {
        ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
        for (S3Object content : listObjResponse.contents()) {
            System.out.println(content.key());
        }

        if (listObjResponse.nextContinuationToken() == null) {
            done = true;
        }

        listObjectsReqManual = listObjectsReqManual.toBuilder()
            .continuationToken(listObjResponse.nextContinuationToken())
            .build();
    }
}

public static void anotherListExample(S3Client s3, String bucketName) {

    ListObjectsV2Request listReq = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);

    // Process response pages
    listRes.stream()
        .flatMap(r -> r.contents().stream())
        .forEach(content -> System.out.println(" Key: " + content.key() + " size = "
+ content.size()));

    // Helper method to work with paginated collection of items directly
    listRes.contents().stream()
        .forEach(content -> System.out.println(" Key: " + content.key() + " size = "
+ content.size()));

    for (S3Object content : listRes.contents()) {
        System.out.println(" Key: " + content.key() + " size = " + content.size());
    }
}

public static void deleteObjectFromBucket(S3Client s3, String bucketName, String
key) {

    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    s3.deleteObject(deleteObjectRequest);
```

```
        System.out.println(key +" was deleted");
    }

    public static String copyBucketObject (S3Client s3, String fromBucket, String
objectKey, String toBucket) {

        String encodedUrl = null;
        try {
            encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
StandardCharsets.UTF_8.toString());
        } catch (UnsupportedEncodingException e) {
            System.out.println("URL could not be encoded: " + e.getMessage());
        }
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .copySource(encodedUrl)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        try {
            CopyObjectResponse copyRes = s3.copyObject(copyReq);
            System.out.println("The "+ objectKey +" was copied to "+toBucket);
            return copyRes.copyObjectResult().toString();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

## S3 Glacier examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon S3 Glacier.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3433\)](#)

## Actions

### Create a vault

The following code example shows how to create an Amazon S3 Glacier vault.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void createGlacierVault(GlacierClient glacier, String vaultName) {  
  
    try {  
        CreateVaultRequest vaultRequest = CreateVaultRequest.builder()  
            .vaultName(vaultName)  
            .build();  
  
        CreateVaultResponse createVaultResult = glacier.createVault(vaultRequest);  
        System.out.println("The URI of the new vault is " +  
            createVaultResult.location());  
  
    } catch(GlacierException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [CreateVault](#) in *AWS SDK for Java 2.x API Reference*.

### Delete a vault

The following code example shows how to delete an Amazon S3 Glacier vault.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteGlacierArchive(GlacierClient glacier, String vaultName,  
String accountId, String archiveId) {  
  
    try {  
        DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()  
            .vaultName(vaultName)  
            .accountId(accountId)  
            .archiveId(archiveId)  
            .build();  
  
        glacier.deleteArchive(delArcRequest);  
        System.out.println("The vault was deleted!");  
  
    } catch(GlacierException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

```
}
```

- For API details, see [DeleteVault](#) in *AWS SDK for Java 2.x API Reference*.

## Delete an archive

The following code example shows how to delete an Amazon S3 Glacier archive.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteGlacierArchive(GlacierClient glacier, String vaultName,
String accountId, String archiveId) {

    try {
        DeleteArchiveRequest delArcRequest = DeleteArchiveRequest.builder()
            .vaultName(vaultName)
            .accountId(accountId)
            .archiveId(archiveId)
            .build();

        glacier.deleteArchive(delArcRequest);
        System.out.println("The vault was deleted!");

    } catch(GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteArchive](#) in *AWS SDK for Java 2.x API Reference*.

## List vaults

The following code example shows how to list Amazon S3 Glacier vaults.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllVault(GlacierClient glacier) {

    boolean listComplete = false;
    String newMarker = null;
    int totalVaults = 0;
    System.out.println("Your Amazon Glacier vaults:");

    try {

        while (!listComplete) {
            ListVaultsResponse response = null;
```

```
if (newMarker != null) {
    ListVaultsRequest request = ListVaultsRequest.builder()
        .marker(newMarker)
        .build();

    response = glacier.listVaults(request);
} else {
    ListVaultsRequest request = ListVaultsRequest.builder()
        .build();
    response = glacier.listVaults(request);
}

List<DescribeVaultOutput> vaultList = response.vaultList();
for (DescribeVaultOutput v: vaultList) {
    totalVaults += 1;
    System.out.println("* " + v.vaultName());
}

// Check for further results.
newMarker = response.marker();
if (newMarker == null) {
    listComplete = true;
}
}

if (totalVaults == 0) {
    System.out.println("No vaults found.");
}

} catch(GlacierException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [ListVaults](#) in *AWS SDK for Java 2.x API Reference*.

## Retrieve a vault inventory

The following code example shows how to retrieve an Amazon S3 Glacier vault inventory.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createJob(GlacierClient glacier, String vaultName, String
accountId) {

    try {

        JobParameters job = JobParameters.builder()
            .type("inventory-retrieval")
            .build();

        InitiateJobRequest initJob = InitiateJobRequest.builder()
            .jobParameters(job)
            .accountId(accountId)
            .vaultName(vaultName)
            .build();
    }
}
```

```
        InitiateJobResponse response = glacier.initiateJob(initJob);
        System.out.println("The job ID is: " +response.jobId());
        System.out.println("The relative URI path of the job is: "
+response.location());
        return response.jobId();

    } catch(GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);

    }
    return "";
}

// Poll S3 Glacier = Polling a Job may take 4-6 hours according to the
Documentation.
public static void checkJob(GlacierClient glacier, String jobId, String name,
String account, String path) {

try{
    boolean finished = false;
    String jobStatus;
    int yy=0;

    while (!finished) {
        DescribeJobRequest jobRequest = DescribeJobRequest.builder()
            .jobId(jobId)
            .accountId(account)
            .vaultName(name)
            .build();

        DescribeJobResponse response = glacier.describeJob(jobRequest);
        jobStatus = response.statusCodeAsString();

        if (jobStatus.compareTo("Succeeded") == 0)
            finished = true;
        else {
            System.out.println(yy + " status is: " + jobStatus);
            Thread.sleep(1000);
        }
        yy++;
    }

    System.out.println("Job has Succeeded");
    GetJobOutputRequest jobOutputRequest = GetJobOutputRequest.builder()
        .jobId(jobId)
        .vaultName(name)
        .accountId(account)
        .build();

    ResponseBytes<GetJobOutputResponse> objectBytes =
glacier.getJobOutputAsBytes(jobOutputRequest);
    // Write the data to a local file.
    byte[] data = objectBytes.asByteArray();
    File myFile = new File(path);
    OutputStream os = new FileOutputStream(myFile);
    os.write(data);
    System.out.println("Successfully obtained bytes from a Glacier vault");
    os.close();

} catch(GlacierException | InterruptedException | IOException e) {
    System.out.println(e.getMessage());
    System.exit(1);

}
```

```
}
```

- For API details, see [InitiateJob](#) in *AWS SDK for Java 2.x API Reference*.

## Upload an archive to a vault

The following code example shows how to upload an archive to an Amazon S3 Glacier vault.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String uploadContent(GlacierClient glacier, Path path, String
vaultName, File myFile) {

    // Get an SHA-256 tree hash value.
    String checkVal = computeSHA256(myFile);
    try {
        UploadArchiveRequest uploadRequest = UploadArchiveRequest.builder()
            .vaultName(vaultName)
            .checksum(checkVal)
            .build();

        UploadArchiveResponse res = glacier.uploadArchive(uploadRequest, path);
        return res.archiveId();

    } catch(GlacierException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

private static String computeSHA256(File inputFile) {

    try {
        byte[] treeHash = computeSHA256TreeHash(inputFile);
        System.out.printf("SHA-256 tree hash = %s\n", toHex(treeHash));
        return toHex(treeHash);

    } catch (IOException ioe) {
        System.err.format("Exception when reading from file %s: %s",
        ioe.getMessage());
        System.exit(-1);

    } catch (NoSuchAlgorithmException nsae) {
        System.err.format("Cannot locate MessageDigest algorithm for SHA-256: %s",
        nsae.getMessage());
        System.exit(-1);
    }
    return "";
}

public static byte[] computeSHA256TreeHash(File inputFile) throws IOException,
NoSuchAlgorithmException {

    byte[][] chunkSHA256Hashes = getChunkSHA256Hashes(inputFile);
    return computeSHA256TreeHash(chunkSHA256Hashes);
}
```

```

/**
 * Computes an SHA256 checksum for each 1 MB chunk of the input file. This
 * includes the checksum for the last chunk, even if it's smaller than 1 MB.
 */
public static byte[][] getChunkSHA256Hashes(File file) throws IOException,
    NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    long numChunks = file.length() / ONE_MB;
    if (file.length() % ONE_MB > 0) {
        numChunks++;
    }

    if (numChunks == 0) {
        return new byte[][] { md.digest() };
    }

    byte[][] chunkSHA256Hashes = new byte[(int) numChunks][];
    FileInputStream fileStream = null;

    try {
        fileStream = new FileInputStream(file);
        byte[] buff = new byte[ONE_MB];

        int bytesRead;
        int idx = 0;

        while ((bytesRead = fileStream.read(buff, 0, ONE_MB)) > 0) {
            md.reset();
            md.update(buff, 0, bytesRead);
            chunkSHA256Hashes[idx++] = md.digest();
        }

        return chunkSHA256Hashes;
    } finally {
        if (fileStream != null) {
            try {
                fileStream.close();
            } catch (IOException ioe) {
                System.err.printf("Exception while closing %s.\n %s",
                    file.getName(),
                    ioe.getMessage());
            }
        }
    }
}

/**
 * Computes the SHA-256 tree hash for the passed array of 1 MB chunk
 * checksums.
 */
public static byte[] computeSHA256TreeHash(byte[][] chunkSHA256Hashes)
    throws NoSuchAlgorithmException {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    byte[][] prevLvlHashes = chunkSHA256Hashes;
    while (prevLvlHashes.length > 1) {
        int len = prevLvlHashes.length / 2;
        if (prevLvlHashes.length % 2 != 0) {
            len++;
        }

        byte[][] currLvlHashes = new byte[len][];
        int j = 0;

```

```
        for (int i = 0; i < prevLvlHashes.length; i = i + 2, j++) {  
  
            // If there are at least two elements remaining.  
            if (prevLvlHashes.length - i > 1) {  
  
                // Calculate a digest of the concatenated nodes.  
                md.reset();  
                md.update(prevLvlHashes[i]);  
                md.update(prevLvlHashes[i + 1]);  
                currLvlHashes[j] = md.digest();  
  
            } else { // Take care of the remaining odd chunk  
                currLvlHashes[j] = prevLvlHashes[i];  
            }  
        }  
  
        prevLvlHashes = currLvlHashes;  
    }  
  
    return prevLvlHashes[0];  
}  
  
/**  
 * Returns the hexadecimal representation of the input byte array  
 */  
public static String toHex(byte[] data) {  
    StringBuilder sb = new StringBuilder(data.length * 2);  
    for (byte datum : data) {  
        String hex = Integer.toHexString(datum & 0xFF);  
  
        if (hex.length() == 1) {  
            // Append leading zero.  
            sb.append("0");  
        }  
        sb.append(hex);  
    }  
    return sb.toString().toLowerCase();  
}
```

- For API details, see [UploadArchive](#) in *AWS SDK for Java 2.x API Reference*.

## Amazon SES examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Simple Email Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3439\)](#)

## Actions

### List email templates

The following code example shows how to list Amazon SES email templates.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllTemplates(SesV2Client sesv2Client) {  
  
    try {  
        ListEmailTemplatesRequest templatesRequest =  
ListEmailTemplatesRequest.builder()  
            .pageSize(1)  
            .build();  
  
        ListEmailTemplatesResponse response =  
sesv2Client.listEmailTemplates(templatesRequest);  
        response.templatesMetadata().forEach(template ->  
            System.out.println("Template name: " + template.templateName()));  
  
    } catch (SesV2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListTemplates](#) in *AWS SDK for Java 2.x API Reference*.

## List identities

The following code example shows how to list Amazon SES identities.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listSESIentities(SesClient client) {  
  
    try {  
        ListIdentitiesResponse identitiesResponse = client.listIdentities();  
        List<String> identities = identitiesResponse.identities();  
        for (String identity: identities) {  
            System.out.println("The identity is "+identity);  
        }  
  
    } catch (SesException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListIdentities](#) in *AWS SDK for Java 2.x API Reference*.

## Send email

The following code example shows how to send email with Amazon SES.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void send(SesClient client,
                      String sender,
                      String recipient,
                      String subject,
                      String bodyHTML)
    ) throws MessagingException {

    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    Content content = Content.builder()
        .data(bodyHTML)
        .build();

    Content sub = Content.builder()
        .data(subject)
        .build();

    Body body = Body.builder()
        .html(content)
        .build();

    Message msg = Message.builder()
        .subject(sub)
        .body(body)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .message(msg)
        .source(sender)
        .build();

    try {
        System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");
        client.sendEmail(emailRequest);

    } catch (SesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void sendemailAttachment(SesClient client,
                                      String sender,
                                      String recipient,
                                      String subject,
                                      String bodyText,
                                      String bodyHTML,
                                      String fileLocation) throws AddressException,
MessagingException, IOException {

    java.io.File theFile = new java.io.File(fileLocation);
    byte[] fileContent = Files.readAllBytes(theFile.toPath());

    Session session = Session.getDefaultInstance(new Properties());
```

```
// Create a new MimeMessage object.
MimeMessage message = new MimeMessage(session);

// Add subject, from and to lines.
message.setSubject(subject, "UTF-8");
message.setFrom(new InternetAddress(sender));
message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(recipient));

// Create a multipart/alternative child container.
MimeMultipart msgBody = new MimeMultipart("alternative");

// Create a wrapper for the HTML and text parts.
MimeBodyPart wrap = new MimeBodyPart();

// Define the text part.
MimeBodyPart textPart = new MimeBodyPart();
textPart.setContent(bodyText, "text/plain; charset=UTF-8");

// Define the HTML part.
MimeBodyPart htmlPart = new MimeBodyPart();
htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");

// Add the text and HTML parts to the child container.
msgBody.addBodyPart(textPart);
msgBody.addBodyPart(htmlPart);

// Add the child container to the wrapper object.
wrap.setContent(msgBody);

// Create a multipart/mixed parent container.
MimeMultipart msg = new MimeMultipart("mixed");

// Add the parent container to the message.
message.setContent(msg);
msg.addBodyPart(wrap);

// Define the attachment.
MimeBodyPart att = new MimeBodyPart();
DataSource fds = new ByteArrayDataSource(fileContent, "application/
vnd.openxmlformats-officedocument.spreadsheetml.sheet");
att.setDataHandler(new DataHandler(fds));

String reportName = "WorkReport.xls";
att.setFileName(reportName);

// Add the attachment to the message.
msg.addBodyPart(att);

try {
    System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");

    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    message.writeTo(outputStream);

    ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());

    byte[] arr = new byte[buf.remaining()];
    buf.get(arr);

    SdkBytes data = SdkBytes.fromByteArray(arr);
    RawMessage rawMessage = RawMessage.builder()
        .data(data)
        .build();
}
```

```
        SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
            .rawMessage(rawMessage)
            .build();

        client.sendRawEmail(rawEmailRequest);

    } catch (SesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Email sent using SesClient with attachment");
}
```

- For API details, see [SendEmail](#) in *AWS SDK for Java 2.x API Reference*.

## Send templated email

The following code example shows how to send templated email with Amazon SES.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void send(SesV2Client client, String sender, String recipient, String
templateName){

    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    /*
     * Specify both name and favorite animal (favoriteanimal) in your code when
     * defining the Template object.
     * If you don't specify all the variables in the template, Amazon SES doesn't
     * send the email.
     */
    Template myTemplate = Template.builder()
        .templateName(templateName)
        .templateData("{\n" +
            "  \"name\": \"Jason\"\n," +
            "  \"favoriteanimal\": \"Cat\"\n" +
            "}")
        .build();

    EmailContent emailContent = EmailContent.builder()
        .template(myTemplate)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .content(emailContent)
        .fromEmailAddress(sender)
        .build();

    try {
        System.out.println("Attempting to send an email based on a template using
the AWS SDK for Java (v2)...");
        client.sendEmail(emailRequest);
        System.out.println("email based on a template was sent");
    }
```

```
        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
    }
}
```

- For API details, see [SendTemplatedEmail](#) in *AWS SDK for Java 2.x API Reference*.

## Amazon SES API v2 examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Simple Email Service API v2.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3444\)](#)

## Actions

### Send an email

The following code example shows how to send an Amazon SES API v2 email.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Sends a message.

```
public static void send(SesV2Client client,
                      String sender,
                      String recipient,
                      String subject,
                      String bodyHTML
){
    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    Content content = Content.builder()
        .data(bodyHTML)
        .build();

    Content sub = Content.builder()
        .data(subject)
        .build();

    Body body = Body.builder()
        .html(content)
        .build();

    Message msg = Message.builder()
```

```
.subject(sub)
.body(body)
.build();

EmailContent emailContent = EmailContent.builder()
.simple(msg)
.build();

SendEmailRequest emailRequest = SendEmailRequest.builder()
.destination(destination)
.content(emailContent)
.fromEmailAddress(sender)
.build();

try {
    System.out.println("Attempting to send an email through Amazon SES " +
"using the AWS SDK for Java...");
    client.sendEmail(emailRequest);
    System.out.println("email was sent");

} catch (SesV2Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [SendEmail](#) in *AWS SDK for Java 2.x API Reference*.

## Amazon SNS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Simple Notification Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3445\)](#)
- [Scenarios \(p. 3456\)](#)

## Actions

### Add tags to a topic

The following code example shows how to add tags to an Amazon SNS topic.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void addTopicTags(SnsClient snsClient, String topicArn) {

    try {
        Tag tag = Tag.builder()
            .key("Team")
```

```
        .value("Development")
        .build();

    Tag tag2 = Tag.builder()
        .key("Environment")
        .value("Gamma")
        .build();

    List<Tag> tagList = new ArrayList<>();
    tagList.add(tag);
    tagList.add(tag2);

    TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
        .resourceArn(topicArn)
        .tags(tagList)
        .build();

    snsClient.tagResource(tagResourceRequest);
    System.out.println("Tags have been added to "+topicArn);

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [TagResource](#) in *AWS SDK for Java 2.x API Reference*.

## Check whether a phone number is opted out

The following code example shows how to check whether a phone number is opted out of receiving Amazon SNS messages.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void checkPhone(SnsClient snsClient, String phoneNumber) {

    try {
        CheckIfPhoneNumberIsOptedOutRequest request =
CheckIfPhoneNumberIsOptedOutRequest.builder()
        .phoneNumber(phoneNumber)
        .build();

        CheckIfPhoneNumberIsOptedOutResponse result =
snsClient.checkIfPhoneNumberIsOptedOut(request);
        System.out.println(result.isOptedOut() + "Phone Number " + phoneNumber + " has Opted Out of receiving sns messages."
            + "\n\nStatus was " + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CheckIfPhoneNumberIsOptedOut](#) in *AWS SDK for Java 2.x API Reference*.

## Confirm an endpoint owner wants to receive messages

The following code example shows how to confirm the owner of an endpoint wants to receive Amazon SNS messages by validating the token sent to the endpoint by an earlier Subscribe action.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void confirmSub(SnsClient snsClient, String subscriptionToken, String topicArn) {  
  
    try {  
        ConfirmSubscriptionRequest request = ConfirmSubscriptionRequest.builder()  
            .token(subscriptionToken)  
            .topicArn(topicArn)  
            .build();  
  
        ConfirmSubscriptionResponse result =  
            snsClient.confirmSubscription(request);  
        System.out.println("\n\nStatus was " +  
            result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n" +  
            result.subscriptionArn());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ConfirmSubscription](#) in *AWS SDK for Java 2.x API Reference*.

## Create a topic

The following code example shows how to create an Amazon SNS topic.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createSNSTopic(SnsClient snsClient, String topicName) {  
  
    CreateTopicResponse result = null;  
    try {  
        CreateTopicRequest request = CreateTopicRequest.builder()  
            .name(topicName)  
            .build();  
  
        result = snsClient.createTopic(request);  
        return result.topicArn();  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";
```

```
}
```

- For API details, see [CreateTopic](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a subscription

The following code example shows how to delete an Amazon SNS subscription.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void unSub(SnsClient snsClient, String subscriptionArn) {  
  
    try {  
        UnsubscribeRequest request = UnsubscribeRequest.builder()  
            .subscriptionArn(subscriptionArn)  
            .build();  
  
        UnsubscribeResponse result = snsClient.unsubscribe(request);  
        System.out.println("\n\nStatus was " +  
result.sdkHttpResponse().statusCode()  
            + "\n\nSubscription was removed for " + request.subscriptionArn());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [Unsubscribe](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a topic

The following code example shows how to delete an Amazon SNS topic and all subscriptions to that topic.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteSNSTopic(SnsClient snsClient, String topicArn ) {  
  
    try {  
        DeleteTopicRequest request = DeleteTopicRequest.builder()  
            .topicArn(topicArn)  
            .build();  
  
        DeleteTopicResponse result = snsClient.deleteTopic(request);  
        System.out.println("\n\nStatus was " +  
result.sdkHttpResponse().statusCode());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
    }  
}
```

```
        System.exit(1);
    }
}
```

- For API details, see [DeleteTopic](#) in *AWS SDK for Java 2.x API Reference*.

## Get the properties of a topic

The following code example shows how to get the properties of an Amazon SNS topic.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getSNSTopicAttributes(SnsClient snsClient, String topicArn ) {

    try {
        GetTopicAttributesRequest request = GetTopicAttributesRequest.builder()
            .topicArn(topicArn)
            .build();

        GetTopicAttributesResponse result = snsClient.getTopicAttributes(request);
        System.out.println("\n\nStatus is " + result.sdkHttpResponse().statusCode()
+ "\n\nAttributes: \n\n" + result.attributes());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for Java 2.x API Reference*.

## Get the settings for sending SMS messages

The following code example shows how to get the settings for sending Amazon SNS SMS messages.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getSNSAttributes(SnsClient snsClient, String topicArn ) {

    try {
        GetSubscriptionAttributesRequest request =
GetSubscriptionAttributesRequest.builder()
            .subscriptionArn(topicArn)
            .build();

        // Get the Subscription attributes
        GetSubscriptionAttributesResponse res =
snsClient.getSubscriptionAttributes(request);
        Map<String, String> map = res.attributes();
```

```
// Iterate through the map
Iterator iter = map.entrySet().iterator();
while (iter.hasNext()) {
    Map.Entry entry = (Map.Entry) iter.next();
    System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
entry.getValue());
}

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("\n\nStatus was good");
}
```

- For API details, see [GetSMSAttributes](#) in *AWS SDK for Java 2.x API Reference*.

### List opted out phone numbers

The following code example shows how to list phone numbers that are opted out of receiving Amazon SNS messages.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void list0pts( SnsClient snsClient ) {

    try {
        ListPhoneNumbersOptedOutRequest request =
ListPhoneNumbersOptedOutRequest.builder().build();
        ListPhoneNumbersOptedOutResponse result =
snsClient.listPhoneNumbersOptedOut(request);
        System.out.println("Status is " + result.sdkHttpResponse().statusCode() +
"\n\nPhone Numbers: \n\n" + result.phoneNumbers());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListPhoneNumbersOptedOut](#) in *AWS SDK for Java 2.x API Reference*.

### List the subscribers of a topic

The following code example shows how to retrieve the list of subscribers of an Amazon SNS topic.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listSNSSubscriptions( SnsClient snsClient ) {
```

```
try {
    ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
        .build();

    ListSubscriptionsResponse result = snsClient.listSubscriptions(request);
    System.out.println(result.subscriptions());

} catch (SnsException e) {

    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

- For API details, see [ListSubscriptions](#) in *AWS SDK for Java 2.x API Reference*.

## List topics

The following code example shows how to list Amazon SNS topics.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listSNSTopics(SnsClient snsClient) {

    try {
        ListTopicsRequest request = ListTopicsRequest.builder()
            .build();

        ListTopicsResponse result = snsClient.listTopics(request);
        System.out.println("Status was " + result.sdkHttpResponse().statusCode() +
"\n\nTopics\n\n" + result.topics());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListTopics](#) in *AWS SDK for Java 2.x API Reference*.

## Publish an SMS text message

The following code example shows how to publish SMS messages using Amazon SNS.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
    try {
```

```
PublishRequest request = PublishRequest.builder()  
    .message(message)  
    .phoneNumber(phoneNumber)  
    .build();  
  
PublishResponse result = snsClient.publish(request);  
System.out.println(result.messageId() + " Message sent. Status was " +  
result.sdkHttpResponse().statusCode());  
  
} catch (SnsException e) {  
    System.err.println(e.awsErrorDetails().errorMessage());  
    System.exit(1);  
}  
}
```

- For API details, see [Publish](#) in *AWS SDK for Java 2.x API Reference*.

## Publish to a topic

The following code example shows how to publish messages to an Amazon SNS topic.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void pubTopic(SnsClient snsClient, String message, String topicArn) {  
  
    try {  
        PublishRequest request = PublishRequest.builder()  
            .message(message)  
            .topicArn(topicArn)  
            .build();  
  
        PublishResponse result = snsClient.publish(request);  
        System.out.println(result.messageId() + " Message sent. Status is " +  
result.sdkHttpResponse().statusCode());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [Publish](#) in *AWS SDK for Java 2.x API Reference*.

## Set a filter policy

The following code example shows how to set an Amazon SNS filter policy.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
```

```
try {
    SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
    // Add a filter policy attribute with a single value
    fp.addAttribute("store", "example_corp");
    fp.addAttribute("event", "order_placed");

    // Add a prefix attribute
    fp.addAttributePrefix("customer_interests", "bas");

    // Add an anything-but attribute
    fp.addAttributeAnythingBut("customer_interests", "baseball");

    // Add a filter policy attribute with a list of values
    ArrayList<String> attributeValues = new ArrayList<>();
    attributeValues.add("rugby");
    attributeValues.add("soccer");
    attributeValues.add("hockey");
    fp.addAttribute("customer_interests", attributeValues);

    // Add a numeric attribute
    fp.addAttribute("price_usd", "=", 0);

    // Add a numeric attribute with a range
    fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

    // Apply the filter policy attributes to an Amazon SNS subscription
    fp.apply(snsClient, subscriptionArn);

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [SetSubscriptionAttributes](#) in *AWS SDK for Java 2.x API Reference*.

## Set the default settings for sending SMS messages

The following code example shows how to set the default settings for sending SMS messages using Amazon SNS.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public class SetSMSAttributes {
    public static void main(String[] args) {

        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket" );

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }
}
```

```
public static void setSNSAttributes( SnsClient snsClient, HashMap<String, String> attributes) {

    try {
        SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
            .attributes(attributes)
            .build();

        SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
        System.out.println("Set default Attributes to " + attributes + ". Status was " + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SetSmsAttributes](#) in *AWS SDK for Java 2.x API Reference*.

## Set topic attributes

The following code example shows how to set Amazon SNS topic attributes.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void setTopAttr(SnsClient snsClient, String attribute, String topicArn, String value) {

    try {
        SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
            .attributeName(attribute)
            .attributeValue(value)
            .topicArn(topicArn)
            .build();

        SetTopicAttributesResponse result = snsClient.setTopicAttributes(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode() + "\n\nTopic " + request.topicArn() +
            " updated " + request.attributeName() + " to " +
request.attributeValue());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [SetTopicAttributes](#) in *AWS SDK for Java 2.x API Reference*.

## Subscribe a Lambda function to a topic

The following code example shows how to subscribe a Lambda function so it receives notifications from an Amazon SNS topic.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String subLambda(SnsClient snsClient, String topicArn, String lambdaArn) {  
  
    try {  
        SubscribeRequest request = SubscribeRequest.builder()  
            .protocol("lambda")  
            .endpoint(lambdaArn)  
            .returnSubscriptionArn(true)  
            .topicArn(topicArn)  
            .build();  
  
        SubscribeResponse result = snsClient.subscribe(request);  
        return result.subscriptionArn();  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [Subscribe](#) in *AWS SDK for Java 2.x API Reference*.

## Subscribe an HTTP endpoint to a topic

The following code example shows how to subscribe an HTTP or HTTPS endpoint so it receives notifications from an Amazon SNS topic.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void subHTTPS(SnsClient snsClient, String topicArn, String url ) {  
  
    try {  
        SubscribeRequest request = SubscribeRequest.builder()  
            .protocol("https")  
            .endpoint(url)  
            .returnSubscriptionArn(true)  
            .topicArn(topicArn)  
            .build();  
  
        SubscribeResponse result = snsClient.subscribe(request);  
        System.out.println("Subscription ARN is " + result.subscriptionArn() + "\n"  
"\n Status is " + result.sdkHttpResponse().statusCode());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [Subscribe](#) in *AWS SDK for Java 2.x API Reference*.

### Subscribe an email address to a topic

The following code example shows how to subscribe an email address to an Amazon SNS topic.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void subEmail(SnsClient snsClient, String topicArn, String email) {  
  
    try {  
        SubscribeRequest request = SubscribeRequest.builder()  
            .protocol("email")  
            .endpoint(email)  
            .returnSubscriptionArn(true)  
            .topicArn(topicArn)  
            .build();  
  
        SubscribeResponse result = snsClient.subscribe(request);  
        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n\n"  
    Status is " + result.sdkHttpResponse().statusCode());  
  
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [Subscribe](#) in *AWS SDK for Java 2.x API Reference*.

## Scenarios

### Create a platform endpoint for push notifications

The following code example shows how to create a platform endpoint for Amazon SNS push notifications.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public class RegistrationExample {  
  
    public static void main(String[] args) {  
  
        final String usage = "\n" +  
            "Usage: " +  
            "    <token>\n\n" +  
            "Where:\n" +  
            "    token - The name of the FIFO topic. \n\n" +  
            "    platformApplicationArn - The ARN value of platform application. You can  
        get this value from the AWS Management Console. \n\n";
```

```
if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String token = args[0];
String platformApplicationArn = args[1];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

createEndpoint(snsClient, token, platformApplicationArn);
}

public static void createEndpoint(SnsClient snsClient, String token, String
platformApplicationArn){

    System.out.println("Creating platform endpoint with token " + token);

    try {
        CreatePlatformEndpointRequest endpointRequest =
CreatePlatformEndpointRequest.builder()
            .token(token)
            .platformApplicationArn(platformApplicationArn)
            .build();

        CreatePlatformEndpointResponse response =
snsClient.createPlatformEndpoint(endpointRequest);
        System.out.println("The ARN of the endpoint is " + response.endpointArn());
    } catch ( SnsException e ) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

## Create and publish to a FIFO topic

The following code example shows how to create and publish to a FIFO Amazon SNS topic.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a FIFO topic and FIFO queues. Subscribe the queues to the topic.

```
public static void main(String[] args) {

    final String usage = "\n" +
        "Usage: " +
        "    <topicArn>\n\n" +
        "Where:\n" +
        "    fifoTopicName - The name of the FIFO topic. \n\n" +
        "    fifoQueueARN - The ARN value of a SQS FIFO queue. You can get this
value from the AWS Management Console. \n\n";

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }
}
```

```
}

String fifoTopicName = "PriceUpdatesTopic3.fifo";
String fifoQueueARN = "arn:aws:sqs:us-east-1:814548047983:MyPriceSQS.fifo";
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

createFIFO(snsClient, fifoTopicName, fifoQueueARN);
}

public static void createFIFO(SnsClient snsClient, String topicName, String
queueARN) {

try {
    // Create a FIFO topic by using the SNS service client.
    Map<String, String> topicAttributes = new HashMap<>();
    topicAttributes.put("FifoTopic", "true");
    topicAttributes.put("ContentBasedDeduplication", "false");

    CreateTopicRequest topicRequest = CreateTopicRequest.builder()
        .name(topicName)
        .attributes(topicAttributes)
        .build();

    CreateTopicResponse response = snsClient.createTopic(topicRequest);
    String topicArn = response.topicArn();
    System.out.println("The topic ARN is"+topicArn);

    // Subscribe to the endpoint by using the SNS service client.
    // Only Amazon SQS FIFO queues can receive notifications from an Amazon SNS
FIFO topic.
    SubscribeRequest subscribeRequest = SubscribeRequest.builder()
        .topicArn(topicArn)
        .endpoint(queueARN)
        .protocol("sqS")
        .build();

    snsClient.subscribe(subscribeRequest);
    System.out.println("The topic is subscribed to the queue.");

    // Compose and publish a message that updates the wholesale price.
    String subject = "Price Update";
    String payload = "{\"product\": 214, \"price\": 79.99}";
    String groupId = "PID-214";
    String dedupId = UUID.randomUUID().toString();
    String attributeName = "business";
    String attributeValue = "wholesale";

    MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
        .dataType("String")
        .stringValue(attributeValue)
        .build();

    Map<String, MessageAttributeValue> attributes = new HashMap<>();
    attributes.put(attributeName, msgAttValue);
    PublishRequest pubRequest = PublishRequest.builder()
        .topicArn(topicArn)
        .subject(subject)
        .message(payload)
        .messageGroupId(groupId)
        .messageDuplicationId(dedupId)
        .messageAttributes(attributes)
        .build();
}
```

```
        snsClient.publish(pubRequest);
        System.out.println("Message was published to "+topicArn);

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

## Publish SMS messages to a topic

The following code example shows how to:

- Create an Amazon SNS topic.
- Subscribe phone numbers to the topic.
- Publish SMS messages to the topic so that all subscribed phone numbers receive the message at once.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a topic and return its ARN.

```
public static String createSNSTopic(SnsClient snsClient, String topicName ) {

    CreateTopicResponse result = null;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

Subscribe an endpoint to a topic.

```
public static void subTextSMS( SnsClient snsClient, String topicArn, String
phoneNumber) {

    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("sms")
            .endpoint(phoneNumber)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
```

```
        System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n\n"
Status is " + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Set attributes on the message, such as the ID of the sender, the maximum price, and its type. Message attributes are optional.

```
public class SetSMSAttributes {
    public static void main(String[] args) {

        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket" );

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();
        setSMSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSMSAttributes( SnsClient snsClient, HashMap<String, String>
attributes) {

        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                .attributes(attributes)
                .build();

            SetSmsAttributesResponse result = snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ". Status
was " + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Publish a message to a topic. The message is sent to every subscriber.

```
public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .phoneNumber(phoneNumber)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out.println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}
```

## Amazon SQS examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Simple Queue Service.

**Actions** are code excerpts that show you how to call individual service functions.

**Scenarios** are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3461\)](#)

## Actions

### Create a queue

The following code example shows how to create an Amazon SQS queue.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createQueue(SqsClient sqsClient, String queueName ) {

    try {
        System.out.println("\nCreate Queue");

        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
            .queueName(queueName)
            .build();

        sqsClient.createQueue(createQueueRequest);

        System.out.println("\nGet queue url");

        GetQueueUrlResponse getQueueUrlResponse =
            sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
        return getQueueUrlResponse.queueUrl();

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void listQueues(SqsClient sqsClient) {

    System.out.println("\nList Queues");
    String prefix = "que";

    try {
```

```
        ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
        ListQueuesResponse listQueuesResponse =
sqscClient.listQueues(listQueuesRequest);
        for (String url : listQueuesResponse.queueUrls()) {
            System.out.println(url);
        }

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listQueuesFilter(SqsClient sqsClient, String queueUrl ) {
// List queues with filters
String namePrefix = "queue";
ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
    .queueNamePrefix(namePrefix)
    .build();

ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
System.out.println("Queue URLs with prefix: " + namePrefix);
for (String url : listQueuesFilteredResponse.queueUrls()) {
    System.out.println(url);
}

System.out.println("\nSend message");
try {
    sqsClient.sendMessage(SendMessageRequest.builder()
        .queueUrl(queueUrl)
        .messageBody("Hello world!")
        .delaySeconds(10)
        .build());

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

System.out.println("\nSend multiple messages");
try {
    SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
    .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg
1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10).build())
    .build();
    sqsClient.sendMessageBatch(sendMessageBatchRequest);

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static List<Message> receiveMessages(SqsClient sqsClient, String queueUrl) {
```

```
System.out.println("\nReceive messages");
try {
    ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
    .queueUrl(queueUrl)
    .maxNumberOfMessages(5)
    .build();
    return sqsClient.receiveMessage(receiveMessageRequest).messages();
} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}

public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

    System.out.println("\nChange Message Visibility");
    try {

        for (Message message : messages) {
            ChangeMessageVisibilityRequest req =
ChangeMessageVisibilityRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .visibilityTimeout(100)
                .build();
            sqsClient.changeMessageVisibility(req);
        }
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
    System.out.println("\nDelete Messages");

    try {
        for (Message message : messages) {
            DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                .queueUrl(queueUrl)
                .receiptHandle(message.receiptHandle())
                .build();
            sqsClient.deleteMessage(deleteMessageRequest);
        }
    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [CreateQueue](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a message from a queue

The following code example shows how to delete a message from an Amazon SQS queue.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
    .queueUrl(queueUrl)
    .receiptHandle(message.receiptHandle())
    .build();
    sqsClient.deleteMessage(deleteMessageRequest);
}
```

- For API details, see [DeleteMessage](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a queue

The following code example shows how to delete an Amazon SQS queue.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {

    try {
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteQueue](#) in *AWS SDK for Java 2.x API Reference*.

## Get the URL of a queue

The following code example shows how to get the URL of an Amazon SQS queue.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    GetQueueUrlResponse getQueueUrlResponse =
    sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
    return getQueueUrlResponse.queueUrl();

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
```

- For API details, see [GetQueueUrl](#) in *AWS SDK for Java 2.x API Reference*.

## List queues

The following code example shows how to list Amazon SQS queues.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
String prefix = "que";

try {
    ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
    ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
    for (String url : listQueuesResponse.queueUrls()) {
        System.out.println(url);
    }

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

- For API details, see [ListQueues](#) in *AWS SDK for Java 2.x API Reference*.

## Receive messages from a queue

The following code example shows how to receive messages from an Amazon SQS queue.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
    ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
    .queueUrl(queueUrl)
    .maxNumberOfMessages(5)
    .build();
    return sqsClient.receiveMessage(receiveMessageRequest).messages();

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
```

- For API details, see [ReceiveMessage in AWS SDK for Java 2.x API Reference](#).

## Send a batch of messages to a queue

The following code example shows how to send a batch of messages to an Amazon SQS queue.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
    .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from msg
1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10).build())
    .build();
    sqsClient.sendMessageBatch(sendMessageBatchRequest);
```

- For API details, see [SendMessageBatch in AWS SDK for Java 2.x API Reference](#).

## Send a message to a queue

The following code example shows how to send a message to an Amazon SQS queue.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public class SendReceiveMessages {
    private static final String QUEUE_NAME = "testQueue" + new Date().getTime();

    public static void main(String[] args) {

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_WEST_2)
```

```
.build();

try {
    CreateQueueRequest request = CreateQueueRequest.builder()
        .queueName(QUEUE_NAME)
        .build();
    CreateQueueResponse createResult = sqsClient.createQueue(request);

    GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
        .queueName(QUEUE_NAME)
        .build();

    String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();

    SendMessageRequest sendMsgRequest = SendMessageRequest.builder()
        .queueUrl(queueUrl)
        .messageBody("hello world")
        .delaySeconds(5)
        .build();
    sqsClient.sendMessage(sendMsgRequest);

    // Send multiple messages to the queue
    SendMessageBatchRequest sendBatchRequest =
SendMessageBatchRequest.builder()
    .queueUrl(queueUrl)
    .entries(
        SendMessageBatchRequestEntry.builder()
            .messageBody("Hello from message 1")
            .id("msg_1")
            .build()
        ,
        SendMessageBatchRequestEntry.builder()
            .messageBody("Hello from message 2")
            .delaySeconds(10)
            .id("msg_2")
            .build())
    .build();
    sqsClient.sendMessageBatch(sendBatchRequest);

    // Receive messages from the queue
    ReceiveMessageRequest receiveRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();
    List<Message> messages =
sqsClient.receiveMessage(receiveRequest).messages();

    // Print out the messages
    for (Message m : messages) {
        System.out.println("\n" +m.body());
    }
} catch (QueueNameExistsException e) {
    throw e;
}
}
```

- For API details, see [SendMessage](#) in *AWS SDK for Java 2.x API Reference*.

## Secrets Manager examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS Secrets Manager.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3468\)](#)

## Actions

### Create a secret

The following code example shows how to create a Secrets Manager secret.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createNewSecret( SecretsManagerClient secretsClient, String
secretName, String secretValue) {

    try {
        CreateSecretRequest secretRequest = CreateSecretRequest.builder()
            .name(secretName)
            .description("This secret was created by the AWS Secret Manager Java
API")
            .secretString(secretValue)
            .build();

        CreateSecretResponse secretResponse =
secretsClient.createSecret(secretRequest);
        return secretResponse.arn();

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateSecret](#) in *AWS SDK for Java 2.x API Reference*.

### Delete a secret

The following code example shows how to delete a Secrets Manager secret.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteSpecificSecret(SecretsManagerClient secretsClient, String
secretName) {
```

```
try {
    DeleteSecretRequest secretRequest = DeleteSecretRequest.builder()
        .secretId(secretName)
        .build();

    secretsClient.deleteSecret(secretRequest);
    System.out.println(secretName + " is deleted.");
}

} catch (SecretsManagerException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- For API details, see [DeleteSecret](#) in *AWS SDK for Java 2.x API Reference*.

## Describe a secret

The following code example shows how to describe a Secrets Manager secret.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeGivenSecret(SecretsManagerClient secretsClient, String
secretName) {

    try {
        DescribeSecretRequest secretRequest = DescribeSecretRequest.builder()
            .secretId(secretName)
            .build();

        DescribeSecretResponse secretResponse =
secretsClient.describeSecret(secretRequest);
        Instant lastChangedDate = secretResponse.lastChangedDate();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
            DateTimeFormatter.ofLocalizedDateTime( FormatStyle.SHORT )
                .withLocale( Locale.US )
                .withZone( ZoneId.systemDefault() );

        formatter.format( lastChangedDate );
        System.out.println("The date of the last change to " + secretResponse.name()
+" is " + lastChangedDate );

    } catch (SecretsManagerException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeSecret](#) in *AWS SDK for Java 2.x API Reference*.

## Get a secret value

The following code example shows how to get a Secrets Manager secret value.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getValue(SecretsManagerClient secretsClient, String secretName) {  
  
    try {  
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()  
            .secretId(secretName)  
            .build();  
  
        GetSecretValueResponse valueResponse =  
            secretsClient.getSecretValue(valueRequest);  
        String secret = valueResponse.secretString();  
        System.out.println(secret);  
  
    } catch (SecretsManagerException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [GetSecretValue](#) in *AWS SDK for Java 2.x API Reference*.

## List secrets

The following code example shows how to list Secrets Manager secrets.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listAllSecrets(SecretsManagerClient secretsClient) {  
    try {  
        ListSecretsResponse secretsResponse = secretsClient.listSecrets();  
        List<SecretListEntry> secrets = secretsResponse.secretList();  
        for (SecretListEntry secret: secrets) {  
            System.out.println("The secret name is "+secret.name());  
            System.out.println("The secret description is "+secret.description());  
        }  
  
    } catch (SecretsManagerException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [ListSecrets](#) in *AWS SDK for Java 2.x API Reference*.

## Put a value in a secret

The following code example shows how to put a value in a Secrets Manager secret.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void updateMySecret(SecretsManagerClient secretsClient, String secretName, String secretValue) {  
  
    try {  
        UpdateSecretRequest secretRequest = UpdateSecretRequest.builder()  
            .secretId(secretName)  
            .secretString(secretValue)  
            .build();  
  
        secretsClient.updateSecret(secretRequest);  
  
    } catch (SecretsManagerException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [PutSecretValue](#) in *AWS SDK for Java 2.x API Reference*.

## Step Functions examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS Step Functions.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3471\)](#)

## Actions

### Create a state machine

The following code example shows how to create a Step Functions state machine.

## SDK for Java 2.x

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createMachine( SfnClient sfnClient, String roleARN, String stateMachineName, String jsonFile) {  
  
    String json = getJSONString(jsonFile);  
    try {  
        CreateStateMachineRequest machineRequest =  
CreateStateMachineRequest.builder()
```

```
.definition(json)
.name(stateMachineName)
.roleArn(roleARN)
.type(StateMachineType.STANDARD)
.build();

CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
return response.stateMachineArn();

} catch (SfnException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

private static String getJSONString(String path) {
try {
    JSONParser parser = new JSONParser();
    JSONObject data = (JSONObject) parser.parse(new FileReader(path));//path to
the JSON file.
    return data.toJSONString();

} catch (IOException | org.json.simple.parser.ParseException e) {
    e.printStackTrace();
}
return "";
}
```

- For API details, see [CreateStateMachine](#) in *AWS SDK for Java 2.x API Reference*.

## Delete a state machine

The following code example shows how to delete a Step Functions state machine.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {

    try {
        DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        sfnClient.deleteStateMachine(deleteStateMachineRequest);
        System.out.println(stateMachineArn +" was successfully deleted.");

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DeleteStateMachine](#) in *AWS SDK for Java 2.x API Reference*.

## List state machine runs

The following code example shows how to list Step Functions state machine runs.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getExeHistory(SfnClient sfnClient, String exeARN) {
    try {
        GetExecutionHistoryRequest historyRequest =
GetExecutionHistoryRequest.builder()
            .executionArn(exeARN)
            .maxResults(10)
            .build();

        GetExecutionHistoryResponse historyResponse =
sfnClient.getExecutionHistory(historyRequest);
        List<HistoryEvent> events = historyResponse.events();
        for (HistoryEvent event: events) {
            System.out.println("The event type is "+event.type().toString());
        }

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListExecutions](#) in *AWS SDK for Java 2.x API Reference*.

## List state machines

The following code example shows how to list Step Functions state machines.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void listMachines(SfnClient sfnClient) {

    try {
        ListStateMachinesResponse response = sfnClient.listStateMachines();
        List<StateMachineListItem> machines = response.stateMachines();
        for (StateMachineListItem machine :machines) {
            System.out.println("The name of the state machine is:
"+machine.name());
            System.out.println("The ARN value is : "+machine.stateMachineArn());
        }

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [ListStateMachines](#) in *AWS SDK for Java 2.x API Reference*.

## Start a state machine run

The following code example shows how to start a Step Functions state machine run.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonFile) {

    String json = getJSONString(jsonFile);
    UUID uuid = UUID.randomUUID();
    String uuidValue = uuid.toString();
    try {

        StartExecutionRequest executionRequest = StartExecutionRequest.builder()
            .input(json)
            .stateMachineArn(stateMachineArn)
            .name(uuidValue)
            .build();

        StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
        return response.executionArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

private static String getJSONString(String path) {

    try {
        JSONParser parser = new JSONParser();
        JSONObject data = (JSONObject) parser.parse(new FileReader(path)); //path to
the JSON file.
        String json = data.toJSONString();
        return json;

    } catch (IOException | org.json.simple.parser.ParseException e) {
        e.printStackTrace();
    }
    return "";
}
```

- For API details, see [StartExecution](#) in *AWS SDK for Java 2.x API Reference*.

## AWS Support examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with AWS Support.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Get started

### Hello AWS Support

The following code examples show how to get started using AWS Support.

#### .NET

##### AWS SDK for .NET

###### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon AWSSupport;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

public static class HelloSupport
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
        // the AWS Support service.
        // Use your AWS profile name, or leave it blank to use the default profile.
        // You must have one of the following AWS Support plans: Business,
        Enterprise On-Ramp, or Enterprise. Otherwise, an exception will be thrown.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices(_ , services) =>
                services.AddAWSService<IAmazonAWSSupport>()
            .Build();

        // Now the client is available for injection.
        var supportClient = host.Services.GetRequiredService<IAmazonAWSSupport>();

        // You can use await and any of the async methods to get a response.
        var response = await supportClient.DescribeServicesAsync();
        Console.WriteLine($"\\tHello AWS Support! There are
{response.Services.Count} services available.");
    }
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for .NET API Reference*.

#### Java

##### SDK for Java 2.x

###### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
```

```
* Before running this Java (v2) code example, set up your development environment,
* including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* In addition, you must have the AWS Business Support Plan to use the AWS Support
Java API. For more information, see:
*
* https://aws.amazon.com/premiumsupport/plans/
*
* This Java example performs the following task:
*
* 1. Gets and displays available services.
*
*
* NOTE: To see multiple operations, see SupportScenario.
*/

```

```
public class HelloSupport {

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
            .region(region)
            .build();

        System.out.println("***** Step 1. Get and display available services.");
        displayServices(supportClient);
    }

    // Return a List that contains a Service name and Category name.
    public static void displayServices(SupportClient supportClient) {
        try {
            DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
            .language("en")
            .build();

            DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service: services) {
                if (index== 11)
                    break;

                System.out.println("The Service name is: "+service.name());

                // Display the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat: categories) {
                    System.out.println("The category name is: "+cat.name());
                }
                index++ ;
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for Java 2.x API Reference*.

Kotlin

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
Before running this Kotlin code example, set up your development environment,  
including your credentials.  
  
For more information, see the following documentation topic:  
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html  
  
In addition, you must have the AWS Business Support Plan to use the AWS Support  
Java API. For more information, see:  
  
https://aws.amazon.com/premiumsupport/plans/  
  
This Kotlin example performs the following task:  
  
1. Gets and displays available services.  
*/  
  
suspend fun main() {  
    displaySomeServices()  
}  
  
// Return a List that contains a Service name and Category name.  
suspend fun displaySomeServices() {  
    val servicesRequest = DescribeServicesRequest {  
        language = "en"  
    }  
  
    SupportClient { region = "us-west-2" }.use { supportClient ->  
        val response = supportClient.describeServices(servicesRequest)  
        println("Get the first 10 services")  
        var index = 1  
  
        response.services?.forEach { service ->  
            if (index == 11) {  
                return@forEach  
            }  
  
            println("The Service name is: " + service.name)  
  
            // Get the categories for this service.  
            service.categories?.forEach { cat ->  
                println("The category name is ${cat.name}")  
                index++  
            }  
        }  
    }  
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for Kotlin API reference*.

## Topics

- [Actions \(p. 3478\)](#)
- [Scenarios \(p. 3484\)](#)

## Actions

### Add a communication to a case

The following code example shows how to add an AWS Support communication with an attachment to a support case.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void addAttachSupportCase(SupportClient supportClient, String caseId,
String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
        .caseId(caseId)
        .attachmentSetId(attachmentSetId)
        .communicationBody("Please refer to attachment for details.")
        .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to an
AWS Support case");
        else
            System.out.println("There was an error adding the communication to an
AWS Support case");

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [AddCommunicationToCase](#) in *AWS SDK for Java 2.x API Reference*.

### Add an attachment to a set

The following code example shows how to add an AWS Support attachment to an attachment set.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String addAttachment(SupportClient supportClient, String fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
            .build();

        AddAttachmentsToSetRequest setRequest =
        AddAttachmentsToSetRequest.builder()
            .attachments(attachment)
            .build();

        AddAttachmentsToSetResponse response =
        supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

    } catch (SupportException | FileNotFoundException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [AddAttachmentsToSet](#) in *AWS SDK for Java 2.x API Reference*.

## Create a case

The following code example shows how to create a new AWS Support case.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createSupportCase(SupportClient supportClient, List<String> sevCatList, String sevLevel) {
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
            .severityCode(sevLevel.toLowerCase())
            .communicationBody("Test issue with "+serviceCode.toLowerCase())
            .subject("Test case, please ignore")
            .language("en")
            .issueType("technical")
            .build();

        CreateCaseResponse response = supportClient.createCase(caseRequest);
        return response.caseId();

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
    }
}
```

```
        System.exit(1);
    }
    return "";
}
```

- For API details, see [CreateCase](#) in *AWS SDK for Java 2.x API Reference*.

## Describe an attachment

The following code example shows how to describe an attachment for an AWS Support case.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeAttachment(SupportClient supportClient, String attachId) {
    try {
        DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
            .attachmentId(attachId)
            .build();

        DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
        System.out.println("The name of the file is
"+response.attachment().fileName());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeAttachment](#) in *AWS SDK for Java 2.x API Reference*.

## Describe cases

The following code example shows how to describe AWS Support cases.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest = DescribeCasesRequest.builder()
```

```
.maxResults(20)
.afterTime(yesterday.toString())
.beforeTime(now.toString())
.build();

DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
List<CaseDetails> cases = response.cases();
for (CaseDetails sinCase: cases) {
    System.out.println("The case status is "+sinCase.status());
    System.out.println("The case Id is "+sinCase.caseId());
    System.out.println("The case subject is "+sinCase.subject());
}

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- For API details, see [DescribeCases](#) in *AWS SDK for Java 2.x API Reference*.

## Describe communications

The following code example shows how to describe AWS Support communications for a case.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String listCommunications(SupportClient supportClient, String caseId)
{
    try {
        String attachId = null;
        DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
        .caseId(caseId)
        .maxResults(10)
        .build();

        DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
        List<Communication> communications = response.communications();
        for (Communication comm: communications) {
            System.out.println("the body is: " + comm.body());

            //Get the attachment id value.
            List<AttachmentDetails> attachments = comm.attachmentSet();
            for (AttachmentDetails detail : attachments) {
                attachId = detail.attachmentId();
            }
        }
        return attachId;

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

```
}
```

- For API details, see [DescribeCommunications](#) in *AWS SDK for Java 2.x API Reference*.

## Describe services

The following code example shows how to describe the list of AWS services.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest = DescribeServicesRequest.builder()
            .language("en")
            .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        String serviceCode = null;
        String catName = null;
        List<String> sevCatList = new ArrayList<>();
        List<Service> services = response.services();

        System.out.println("Get the first 10 services");
        int index = 1;
        for (Service service: services) {
            if (index== 11)
                break;

            System.out.println("The Service name is: "+service.name());
            if (service.name().compareTo("Account") == 0)
                serviceCode = service.code();

            // Get the Categories for this service.
            List<Category> categories = service.categories();
            for (Category cat: categories) {
                System.out.println("The category name is: "+cat.name());
                if (cat.name().compareTo("Security") == 0)
                    catName = cat.name();
            }
            index++ ;
        }

        // Push the two values to the list.
        sevCatList.add(serviceCode);
        sevCatList.add(catName);
        return sevCatList;
    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return null;
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for Java 2.x API Reference*.

## Describe severity levels

The following code example shows how to describe AWS Support severity levels.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String displaySevLevels(SupportClient supportClient) {  
    try {  
        DescribeSeverityLevelsRequest severityLevelsRequest =  
DescribeSeverityLevelsRequest.builder()  
            .language("en")  
            .build();  
  
        DescribeSeverityLevelsResponse response =  
supportClient.describeSeverityLevels(severityLevelsRequest);  
        List<SeverityLevel> severityLevels = response.severityLevels();  
        String levelName = null;  
        for (SeverityLevel sevLevel: severityLevels) {  
            System.out.println("The severity level name is: " + sevLevel.name());  
            if (sevLevel.name().compareTo("High")==0)  
                levelName = sevLevel.name();  
        }  
        return levelName;  
  
    } catch (SupportException e) {  
        System.out.println(e.getLocalizedMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- For API details, see [DescribeSeverityLevels](#) in *AWS SDK for Java 2.x API Reference*.

## Resolve case

The following code example shows how to resolve an AWS Support case.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void resolveSupportCase(SupportClient supportClient, String caseId) {  
    try {  
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()  
            .caseId(caseId)  
            .build();  
  
        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);  
        System.out.println("The status of case "+caseId+" is  
"+response.finalCaseStatus());  
    }
```

```
        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
    }
}
```

- For API details, see [ResolveCase](#) in *AWS SDK for Java 2.x API Reference*.

## Scenarios

### Get started with cases

The following code example shows how to:

- Get and display available services and severity levels for cases.
- Create a support case using a selected service, category, and severity level.
- Get and display a list of open cases for the current day.
- Add an attachment set and a communication to the new case.
- Describe the new attachment and communication for the case.
- Resolve the case.
- Get and display a list of resolved cases for the current day.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run various AWS Support operations.

```
/**  
 * Before running this Java (v2) code example, set up your development environment,  
 * including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 *  
 * In addition, you must have the AWS Business Support Plan to use the AWS Support  
 * Java API. For more information, see:  
 *  
 * https://aws.amazon.com/premiumsupport/plans/  
 *  
 * This Java example performs the following tasks:  
 *  
 * 1. Gets and displays available services.  
 * 2. Gets and displays severity levels.  
 * 3. Creates a support case by using the selected service, category, and severity  
 * level.  
 * 4. Gets a list of open cases for the current day.  
 * 5. Creates an attachment set with a generated file.  
 * 6. Adds a communication with the attachment to the support case.  
 * 7. Lists the communications of the support case.  
 * 8. Describes the attachment set included with the communication.  
 * 9. Resolves the support case.  
 * 10. Gets a list of resolved cases for the current day.  
 */
```

```
public class SupportScenario {  
  
    public static final String DASHES = new String(new char[80]).replace("\0", "-");  
    public static void main(String[] args) {  
        final String usage = "\n" +  
            "Usage:\n" +  
            "  <fileAttachment>" +  
            "Where:\n" +  
            "  fileAttachment - The file can be a simple saved .txt file to use as an  
email attachment. \n";  
  
        if (args.length != 1) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String fileAttachment = args[0];  
        Region region = Region.US_WEST_2;  
        SupportClient supportClient = SupportClient.builder()  
            .region(region)  
            .build();  
  
        System.out.println(DASHES);  
        System.out.println("***** Welcome to the AWS Support case example scenario.");  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("1. Get and display available services.");  
        List<String> sevCatList = displayServices(supportClient);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("2. Get and display Support severity levels.");  
        String sevLevel = displaySevLevels(supportClient);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("3. Create a support case using the selected service,  
category, and severity level.");  
        String caseId = createSupportCase(supportClient, sevCatList, sevLevel);  
        if (caseId.compareTo("")==0) {  
            System.out.println("A support case was not successfully created!");  
            System.exit(1);  
        } else  
            System.out.println("Support case "+caseId +" was successfully created!");  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("4. Get open support cases.");  
        getOpenCase(supportClient);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("5. Create an attachment set with a generated file to add to  
the case.");  
        String attachmentSetId = addAttachment(supportClient, fileAttachment);  
        System.out.println("The Attachment Set id value is" +attachmentSetId);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("6. Add communication with the attachment to the support  
case.");  
        addAttachSupportCase(supportClient, caseId, attachmentSetId);  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);
```

```
System.out.println("7. List the communications of the support case.");
String attachId = listCommunications(supportClient, caseId);
System.out.println("The Attachment id value is" +attachId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Describe the attachment set included with the
communication.");
describeAttachment(supportClient, attachId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Resolve the support case.");
resolveSupportCase(supportClient, caseId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a list of resolved cases for the current day.");
getResolvedCase(supportClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("***** This Scenario has successfully completed");
System.out.println(DASHES);
}

public static void getResolvedCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest = DescribeCasesRequest.builder()
            .maxResults(30)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
            .includeResolvedCases(true)
            .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase: cases) {
            if (sinCase.status().compareTo("resolved") ==0)
                System.out.println("The case status is "+sinCase.status());
        }
    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void resolveSupportCase(SupportClient supportClient, String caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case "+caseId +" is
"+response.finalCaseStatus());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
```

```
        System.exit(1);
    }

    public static void describeAttachment(SupportClient supportClient, String attachId)
{
    try {
        DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
            .attachmentId(attachId)
            .build();

        DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
        System.out.println("The name of the file is
"+response.attachment().fileName());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static String listCommunications(SupportClient supportClient, String caseId)
{
    try {
        String attachId = null;
        DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
            .caseId(caseId)
            .maxResults(10)
            .build();

        DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
        List<Communication> communications = response.communications();
        for (Communication comm: communications) {
            System.out.println("the body is: " + comm.body());

            //Get the attachment id value.
            List<AttachmentDetails> attachments = comm.attachmentSet();
            for (AttachmentDetails detail : attachments) {
                attachId = detail.attachmentId();
            }
        }
        return attachId;
    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static void addAttachSupportCase(SupportClient supportClient, String caseId,
String attachmentSetId) {
    try {
        AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
            .caseId(caseId)
            .attachmentSetId(attachmentSetId)
            .communicationBody("Please refer to attachment for details.")
            .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
```

```
        if (response.result())
            System.out.println("You have successfully added a communication to an
AWS Support case");
        else
            System.out.println("There was an error adding the communication to an
AWS Support case");

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
            .build();

        AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
            .attachments(attachment)
            .build();

        AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

    } catch (SupportException | FileNotFoundException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest = DescribeCasesRequest.builder()
            .maxResults(20)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
            .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase: cases) {
            System.out.println("The case status is "+sinCase.status());
            System.out.println("The case Id is "+sinCase.caseId());
            System.out.println("The case subject is "+sinCase.subject());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```

        }

    public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
            .severityCode(sevLevel.toLowerCase())
            .communicationBody("Test issue with "+serviceCode.toLowerCase())
            .subject("Test case, please ignore")
            .language("en")
            .issueType("technical")
            .build();

        CreateCaseResponse response = supportClient.createCase(caseRequest);
        return response.caseId();
    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static String displaySevLevels(SupportClient supportClient) {
    try {
        DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
            .language("en")
            .build();

        DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
        List<SeverityLevel> severityLevels = response.severityLevels();
        String levelName = null;
        for (SeverityLevel sevLevel: severityLevels) {
            System.out.println("The severity level name is: " + sevLevel.name());
            if (sevLevel.name().compareTo("High")==0)
                levelName = sevLevel.name();
        }
        return levelName;
    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest = DescribeServicesRequest.builder()
            .language("en")
            .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        String serviceCode = null;
        String catName = null;
        List<String> sevCatList = new ArrayList<>();
        List<Service> services = response.services();

```

```
System.out.println("Get the first 10 services");
int index = 1;
for (Service service: services) {
    if (index== 11)
        break;

    System.out.println("The Service name is: "+service.name());
    if (service.name().compareTo("Account") == 0)
        serviceCode = service.code();

    // Get the Categories for this service.
    List<Category> categories = service.categories();
    for (Category cat: categories) {
        System.out.println("The category name is: "+cat.name());
        if (cat.name().compareTo("Security") == 0)
            catName = cat.name();
    }
    index++ ;
}

// Push the two values to the list.
sevCatList.add(serviceCode);
sevCatList.add(catName);
return sevCatList;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return null;
}
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - [AddAttachmentsToSet](#)
  - [AddCommunicationToCase](#)
  - [CreateCase](#)
  - [DescribeAttachment](#)
  - [DescribeCases](#)
  - [DescribeCommunications](#)
  - [DescribeServices](#)
  - [DescribeSeverityLevels](#)
  - [ResolveCase](#)

## Amazon Textract examples using SDK for Java 2.x

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Java 2.x with Amazon Textract.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3491\)](#)

## Actions

### Analyze a document

The following code example shows how to analyze a document using Amazon Textract.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void analyzeDoc(TextractClient textractClient, String sourceDoc) {  
  
    try {  
        InputStream sourceStream = new FileInputStream(new File(sourceDoc));  
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);  
  
        // Get the input Document object as bytes  
        Document myDoc = Document.builder()  
            .bytes(sourceBytes)  
            .build();  
  
        List<FeatureType> featureTypes = new ArrayList<FeatureType>();  
        featureTypes.add(FeatureType.FORMS);  
        featureTypes.add(FeatureType.TABLES);  
  
        AnalyzeDocumentRequest analyzeDocumentRequest =  
        AnalyzeDocumentRequest.builder()  
            .featureTypes(featureTypes)  
            .document(myDoc)  
            .build();  
  
        AnalyzeDocumentResponse analyzeDocument =  
        textractClient.analyzeDocument(analyzeDocumentRequest);  
        List<Block> docInfo = analyzeDocument.blocks();  
        Iterator<Block> blockIterator = docInfo.iterator();  
  
        while(blockIterator.hasNext()) {  
            Block block = blockIterator.next();  
            System.out.println("The block type is " +block.blockType().toString());  
        }  
  
    } catch (TextractException | FileNotFoundException e) {  
  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- For API details, see [AnalyzeDocument](#) in *AWS SDK for Java 2.x API Reference*.

### Detect text in a document

The following code example shows how to detect text in a document using Amazon Textract.

#### SDK for Java 2.x

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Detect text from an input document.

```
public static void detectDocText(TextractClient textractClient, String sourceDoc) {  
  
    try {  
        InputStream sourceStream = new FileInputStream(new File(sourceDoc));  
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);  
  
        // Get the input Document object as bytes  
        Document myDoc = Document.builder()  
            .bytes(sourceBytes)  
            .build();  
  
        DetectDocumentTextRequest detectDocumentTextRequest =  
        DetectDocumentTextRequest.builder()  
            .document(myDoc)  
            .build();  
  
        // Invoke the Detect operation  
        DetectDocumentTextResponse textResponse =  
        textractClient.detectDocumentText(detectDocumentTextRequest);  
        List<Block> docInfo = textResponse.blocks();  
        for (Block block : docInfo) {  
            System.out.println("The block type is " +  
                block.blockType().toString());  
        }  
  
        DocumentMetadata documentMetadata = textResponse.documentMetadata();  
        System.out.println("The number of pages in the document is "  
            +documentMetadata.pages());  
  
    } catch (TextractException | FileNotFoundException e) {  
  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

Detect text from a document located in an Amazon S3 bucket.

```
public static void detectDocTextS3 (TextractClient textractClient, String  
bucketName, String docName) {  
  
    try {  
        S3Object s3Object = S3Object.builder()  
            .bucket(bucketName)  
            .name(docName)  
            .build();  
  
        // Create a Document object and reference the s3Object instance  
        Document myDoc = Document.builder()  
            .s3Object(s3Object)  
            .build();  
  
        DetectDocumentTextRequest detectDocumentTextRequest =  
        DetectDocumentTextRequest.builder()  
            .document(myDoc)  
            .build();  
  
        DetectDocumentTextResponse textResponse =  
        textractClient.detectDocumentText(detectDocumentTextRequest);  
        for (Block block: textResponse.blocks()) {  
            System.out.println("The block type is " +block.blockType().toString());  
        }  
    } catch (TextractException | FileNotFoundException e) {  
  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

```
        }

        DocumentMetadata documentMetadata = textResponse.documentMetadata();
        System.out.println("The number of pages in the document is "
+documentMetadata.pages());

    } catch (TextractException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- For API details, see [DetectDocumentText](#) in *AWS SDK for Java 2.x API Reference*.

## Start asynchronous analysis of a document

The following code example shows how to start asynchronous analysis of a document using Amazon Textract.

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String startDocAnalysisS3 (TextractClient textractClient, String
bucketName, String docName) {

    try {
        List<FeatureType> myList = new ArrayList<>();
        myList.add(FeatureType.TABLES);
        myList.add(FeatureType.FORMS);

        S3Object s3Object = S3Object.builder()
            .bucket(bucketName)
            .name(docName)
            .build();

        DocumentLocation location = DocumentLocation.builder()
            .s3Object(s3Object)
            .build();

        StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
            .documentLocation(location)
            .featureTypes(myList)
            .build();

        StartDocumentAnalysisResponse response =
textractClient.startDocumentAnalysis(documentAnalysisRequest);

        // Get the job ID
        String jobId = response.jobId();
        return jobId;

    } catch (TextractException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

```
}

private static String getJobResults(TextractClient textractClient, String jobId) {

    boolean finished = false;
    int index = 0 ;
    String status = "" ;

    try {
        while (!finished) {
            GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                .jobId(jobId)
                .maxResults(1000)
                .build();

            GetDocumentAnalysisResponse response =
textractClient.getDocumentAnalysis(analysisRequest);
            status = response.jobStatus().toString();

            if (status.compareTo("SUCCEEDED") == 0)
                finished = true;
            else {
                System.out.println(index + " status is: " + status);
                Thread.sleep(1000);
            }
            index++ ;
        }

        return status;
    } catch( InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- For API details, see [StartDocumentAnalysis](#) in *AWS SDK for Java 2.x API Reference*.

## Cross-service examples using SDK for Java 2.x

The following sample applications use the AWS SDK for Java 2.x to work across multiple AWS services.

### Examples

- [Build an application to submit data to a DynamoDB table \(p. 3495\)](#)
- [Create an Amazon Lex Chatbot within a web application to engage your web site visitors \(p. 3495\)](#)
- [Build a publish and subscription application that translates messages \(p. 3495\)](#)
- [Create a web application that sends and retrieves messages by using the Amazon Simple Queue Service API \(p. 3496\)](#)
- [Create a web application to track DynamoDB data \(p. 3496\)](#)
- [Create an Amazon Redshift item tracker \(p. 3496\)](#)
- [Create an Aurora Serverless work item tracker \(p. 3496\)](#)
- [Detect PPE in images with Amazon Rekognition using an AWS SDK \(p. 3497\)](#)
- [Detect objects in images with Amazon Rekognition using an AWS SDK \(p. 3497\)](#)
- [Detect people and objects in a video with Amazon Rekognition using an AWS SDK \(p. 3497\)](#)

- [Use API Gateway to invoke a Lambda function \(p. 3498\)](#)
- [Use Step Functions to invoke Lambda functions \(p. 3498\)](#)
- [Use scheduled events to invoke a Lambda function \(p. 3498\)](#)

## Build an application to submit data to a DynamoDB table

### SDK for Java 2.x

Shows how to create a dynamic web application that submits data using the Amazon DynamoDB Java API and sends a text message using the Amazon Simple Notification Service Java API.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Amazon SNS

## Create an Amazon Lex Chatbot within a web application to engage your web site visitors

### SDK for Java 2.x

Shows how to use the Amazon Lex API to create a Chatbot within a web application to engage your web site visitors.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

## Build a publish and subscription application that translates messages

### SDK for Java 2.x

Shows how to use the Amazon Simple Notification Service Java API to create a web application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

For complete source code and instructions on how to set up and run the example that uses the Java Async API, see the full example on [GitHub](#).

#### Services used in this example

- Amazon SNS
- Amazon Translate

## Create a web application that sends and retrieves messages by using the Amazon Simple Queue Service API

### SDK for Java 2.x

Shows how to use the Amazon Simple Queue Service API to develop a Spring REST API that sends and retrieves messages.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Comprehend
- Amazon SQS

## Create a web application to track DynamoDB data

### SDK for Java 2.x

Shows how to use the Amazon DynamoDB API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon SES

## Create an Amazon Redshift item tracker

### SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon Redshift database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Redshift data and for use by a React application, see the full example on [GitHub](#).

### Services used in this example

- Amazon Redshift
- Amazon SES

## Create an Aurora Serverless work item tracker

### SDK for Java 2.x

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

For complete source code and instructions on how to set up and run an example that uses the JDBC API, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

## Detect PPE in images with Amazon Rekognition using an AWS SDK

### SDK for Java 2.x

Shows how to create an AWS Lambda function that detects images with Personal Protective Equipment.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

## Detect objects in images with Amazon Rekognition using an AWS SDK

### SDK for Java 2.x

Shows how to use Amazon Rekognition Java API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Detect people and objects in a video with Amazon Rekognition using an AWS SDK

### SDK for Java 2.x

Shows how to use Amazon Rekognition Java API to create an app to detect faces and objects in videos located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Use API Gateway to invoke a Lambda function

### SDK for Java 2.x

Shows how to create an AWS Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create a Lambda function invoked by Amazon API Gateway that scans an Amazon DynamoDB table for work anniversaries and uses Amazon Simple Notification Service (Amazon SNS) to send a text message to your employees that congratulates them at their one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

## Use Step Functions to invoke Lambda functions

### SDK for Java 2.x

Shows how to create an AWS serverless workflow by using AWS Step Functions and the AWS SDK for Java 2.x. Each workflow step is implemented using an AWS Lambda function.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Lambda
- Amazon SES
- Step Functions

## Use scheduled events to invoke a Lambda function

### SDK for Java 2.x

Shows how to create an Amazon EventBridge scheduled event that invokes an AWS Lambda function. Configure EventBridge to use a cron expression to schedule when the Lambda function is invoked. In this example, you create a Lambda function by using the Lambda Java runtime API. This example invokes different AWS services to perform a specific use case. This example demonstrates how to create an app that sends a mobile text message to your employees that congratulates them at the one year anniversary date.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

## Code examples for SDK for Kotlin

The code examples in this topic show you how to use the AWS SDK for Kotlin with AWS.

### Examples

- [Single-service actions and scenarios using SDK for Kotlin \(p. 3499\)](#)
- [Cross-service examples using SDK for Kotlin \(p. 3663\)](#)

## Single-service actions and scenarios using SDK for Kotlin

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with AWS services.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Services

- [Aurora examples using SDK for Kotlin \(p. 3500\)](#)
- [CloudWatch examples using SDK for Kotlin \(p. 3516\)](#)
- [CloudWatch Logs examples using SDK for Kotlin \(p. 3519\)](#)
- [Amazon Cognito Identity Provider examples using SDK for Kotlin \(p. 3520\)](#)
- [DynamoDB examples using SDK for Kotlin \(p. 3530\)](#)
- [Amazon EC2 Auto Scaling examples using SDK for Kotlin \(p. 3548\)](#)
- [EventBridge examples using SDK for Kotlin \(p. 3559\)](#)
- [AWS Glue examples using SDK for Kotlin \(p. 3561\)](#)
- [IAM examples using SDK for Kotlin \(p. 3568\)](#)
- [AWS KMS examples using SDK for Kotlin \(p. 3581\)](#)
- [Lambda examples using SDK for Kotlin \(p. 3587\)](#)
- [Amazon Pinpoint examples using SDK for Kotlin \(p. 3593\)](#)
- [Amazon RDS examples using SDK for Kotlin \(p. 3599\)](#)
- [Amazon Redshift examples using SDK for Kotlin \(p. 3609\)](#)
- [Amazon Rekognition examples using SDK for Kotlin \(p. 3612\)](#)
- [Amazon S3 examples using SDK for Kotlin \(p. 3623\)](#)
- [Amazon SNS examples using SDK for Kotlin \(p. 3632\)](#)
- [Amazon SQS examples using SDK for Kotlin \(p. 3639\)](#)
- [Secrets Manager examples using SDK for Kotlin \(p. 3643\)](#)
- [Step Functions examples using SDK for Kotlin \(p. 3646\)](#)
- [AWS Support examples using SDK for Kotlin \(p. 3649\)](#)

## Aurora examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Aurora.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3500\)](#)
- [Scenarios \(p. 3509\)](#)

## Actions

### Create a DB cluster

The following code example shows how to create an Aurora DB cluster.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createDBCluster(dbParameterGroupFamilyVal: String?, dbName: String?, dbClusterIdentifierVal: String?, userName: String?, password: String?): String? {
    val clusterRequest = CreateDbClusterRequest {
        databaseName = dbName
        dbClusterIdentifier = dbClusterIdentifierVal
        dbClusterParameterGroupName = dbParameterGroupFamilyVal
        engine = "aurora-mysql"
        masterUsername = userName
        masterUserPassword = password
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- For API details, see [CreateDBCluster in AWS SDK for Kotlin API reference](#).

### Create a DB cluster parameter group

The following code example shows how to create an Aurora DB cluster parameter group.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createDBClusterParameterGroup(dbClusterGroupNameVal: String?,  
dbParameterGroupFamilyVal: String?) {  
    val groupRequest = CreateDbClusterParameterGroupRequest {  
        dbClusterParameterGroupName = dbClusterGroupNameVal  
        dbParameterGroupFamily = dbParameterGroupFamilyVal  
        description = "Created by using the AWS SDK for Kotlin"  
    }  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)  
        println("The group name is  
${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")  
    }  
}
```

- For API details, see [CreateDBClusterParameterGroup](#) in *AWS SDK for Kotlin API reference*.

## Create a DB cluster snapshot

The following code example shows how to create an Aurora DB cluster snapshot.

### SDK for Kotlin

**Note**

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createDBClusterSnapshot(dbInstanceClusterIdentifier: String?,  
dbSnapshotIdentifier: String?) {  
    val snapshotRequest = CreateDbClusterSnapshotRequest {  
        dbClusterIdentifier = dbInstanceClusterIdentifier  
        dbClusterSnapshotIdentifier = dbSnapshotIdentifier  
    }  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)  
        println("The Snapshot ARN is  
${response.dbClusterSnapshot?.dbClusterSnapshotArn}")  
    }  
}
```

- For API details, see [CreateDBClusterSnapshot](#) in *AWS SDK for Kotlin API reference*.

## Create a DB instance in a DB cluster

The following code example shows how to create a DB instance in an Aurora DB cluster.

### SDK for Kotlin

**Note**

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createDBInstanceCluster(dbInstanceIdentifierVal: String?,  
    dbInstanceClusterIdentifierVal: String?, instanceClassVal: String?): String? {  
    val instanceRequest = CreateDbInstanceRequest {  
        dbInstanceIdentifier = dbInstanceIdentifierVal  
        dbClusterIdentifier = dbInstanceClusterIdentifierVal  
        engine = "aurora-mysql"  
        dbInstanceClass = instanceClassVal  
    }  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val response = rdsClient.createDbInstance(instanceRequest)  
        print("The status is ${response.dbInstance?.dbInstanceState}")  
        return response.dbInstance?.dbInstanceArn  
    }  
}
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for Kotlin API reference*.

## Delete a DB cluster

The following code example shows how to delete an Aurora DB cluster.

### SDK for Kotlin

**Note**

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {  
    val deleteDbClusterRequest = DeleteDbClusterRequest {  
        dbClusterIdentifier = dbInstanceClusterIdentifier  
        skipFinalSnapshot = true  
    }  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        rdsClient.deleteDbCluster(deleteDbClusterRequest)  
        println("$dbInstanceClusterIdentifier was deleted!")  
    }  
}
```

- For API details, see [DeleteDBCluster](#) in *AWS SDK for Kotlin API reference*.

## Delete a DB cluster parameter group

The following code example shows how to delete an Aurora DB cluster parameter group.

### SDK for Kotlin

**Note**

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
@Throws(InterruptedIOException::class)
suspend fun deleteDBClusterGroup(dbClusterGroupName: String, clusterDBARN: String) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the database
ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest = DeleteDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupName
        }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}
```

- For API details, see [DeleteDBClusterParameterGroup](#) in *AWS SDK for Kotlin API reference*.

## Delete a DB instance

The following code example shows how to delete an Aurora DB instance.

### SDK for Kotlin

**Note**

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest = DeleteDbInstanceRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
        deleteAutomatedBackups = true
        skipFinalSnapshot = true
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is ${response.dbInstanceState}")
    }
}
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for Kotlin API reference*.

## Describe DB cluster parameter groups

The following code example shows how to describe Aurora DB cluster parameter groups.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest = DescribeDbClusterParameterGroupsRequest {
        dbClusterParameterGroupName = dbClusterGroupName
        maxRecords = 20
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

- For API details, see [DescribeDBClusterParameterGroups](#) in *AWS SDK for Kotlin API reference*.

## Describe DB cluster snapshots

The following code example shows how to describe Aurora DB cluster snapshots.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun waitSnapshotReady(dbSnapshotIdentifier: String?,  
    dbInstanceClusterIdentifier: String?) {  
    var snapshotReady = false  
    var snapshotReadyStr: String  
    println("Waiting for the snapshot to become available.")  
  
    val snapshotsRequest = DescribeDbClusterSnapshotsRequest {  
        dbClusterSnapshotIdentifier = dbSnapshotIdentifier  
        dbClusterIdentifier = dbInstanceClusterIdentifier  
    }  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        while (!snapshotReady) {  
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)  
            val snapshotList = response.dbClusterSnapshots  
            if (snapshotList != null) {  
                for (snapshot in snapshotList) {  
                    snapshotReadyStr = snapshot.status.toString()  
                    if (snapshotReadyStr.contains("available")) {  
                        snapshotReady = true  
                    } else {  
                        println(".")  
                        delay(slTime * 5000)  
                    }  
                }  
            }  
        }  
        println("The Snapshot is available!")  
    }  
}
```

- For API details, see [DescribeDBClusterSnapshots](#) in *AWS SDK for Kotlin API reference*.

## Describe DB clusters

The following code example shows how to describe Aurora DB clusters.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeDbClusterParameters(dbClusterGroupName: String?, flag: Int) {  
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest  
    dbParameterGroupsRequest = if (flag == 0) {  
        DescribeDbClusterParametersRequest {  
            dbClusterParameterGroupName = dbClusterGroupName  
        }  
    } else {  
        DescribeDbClusterParametersRequest {  
            dbClusterParameterGroupName = dbClusterGroupName  
            source = "user"  
        }  
    }  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val response = rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
```

```
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 || paraName.compareTo("auto_increment_increment") == 0) {
                    println("**** The parameter name is $paraName")
                    println("**** The parameter value is ${para.parameterValue}")
                    println("**** The parameter data type is ${para.dataType}")
                    println("**** The parameter description is ${para.description}")
                    println("**** The parameter allowed values is
${para.allowedValues}")
                }
            }
        }
    }
```

- For API details, see [DescribeDBClusters](#) in *AWS SDK for Kotlin API reference*.

## Describe DB instances

The following code example shows how to describe Aurora DB instances.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest = DescribeDbInstancesRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
    }

    var endpoint = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceState.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for Kotlin API reference*.

## Describe database engine versions

The following code example shows how to describe Aurora database engine versions.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Get a list of allowed engine versions.  
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {  
    val versionsRequest = DescribeDbEngineVersionsRequest {  
        dbParameterGroupFamily = dbParameterGroupFamilyVal  
        engine = "aurora-mysql"  
    }  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val response = rdsClient.describeDbEngineVersions(versionsRequest)  
        response.dbEngineVersions?.forEach { dbEngine ->  
            println("The engine version is ${dbEngine.engineVersion}")  
            println("The engine description is ${dbEngine.dbEngineDescription}")  
        }  
    }  
}
```

- For API details, see [DescribeDBEngineVersions](#) in *AWS SDK for Kotlin API reference*.

## Describe parameters from a DB cluster parameter group

The following code example shows how to describe parameters from an Aurora DB cluster parameter group.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeDbClusterParameters(dbClusterGroupName: String?, flag: Int) {  
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest  
    dbParameterGroupsRequest = if (flag == 0) {  
        DescribeDbClusterParametersRequest {  
            dbClusterParameterGroupName = dbClusterGroupName  
        }  
    } else {  
        DescribeDbClusterParametersRequest {  
            dbClusterParameterGroupName = dbClusterGroupName  
            source = "user"  
        }  
    }  
}
```

```
RdsClient { region = "us-west-2" }.use { rdsClient ->
    val response = rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
    response.parameters?.forEach { para ->
        // Only print out information about either auto_increment_offset or
        // auto_increment_incremetn.
        val paraName = para.parameterName
        if (paraName != null) {
            if (paraName.compareTo("auto_increment_offset") == 0 || paraName.compareTo("auto_increment_incremetn") == 0) {
                println("**** The parameter name is $paraName")
                println("**** The parameter value is ${para.parameterValue}")
                println("**** The parameter data type is ${para.dataType}")
                println("**** The parameter description is ${para.description}")
                println("**** The parameter allowed values is
${para.allowedValues}")
            }
        }
    }
}
```

- For API details, see [DescribeDBClusterParameters](#) in *AWS SDK for Kotlin API reference*.

## Update parameters in a DB cluster parameter group

The following code example shows how to update parameters in an Aurora DB cluster parameter group.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 = Parameter {
        parameterName = "auto_increment_offset"
        applyMethod = ApplyMethod.fromValue("immediate")
        parameterValue = "5"
    }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest = ModifyDbClusterParameterGroupRequest {
        dbClusterParameterGroupName = dClusterGroupName
        parameters = paraList
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
successfully modified")
    }
}
```

- For API details, see [ModifyDBClusterParameterGroup](#) in *AWS SDK for Kotlin API reference*.

## Scenarios

### Get started with DB clusters

The following code example shows how to:

- Create a custom Aurora DB cluster parameter group and set parameter values.
- Create a DB cluster that is configured to use the parameter group.
- Create a DB instance in the DB cluster that contains a database.
- Take a snapshot of the DB cluster.
- Delete the instance, DB cluster, and parameter group.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
Before running this Kotlin code example, set up your development environment, including  
your credentials.  
  
For more information, see the following documentation topic:  
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html  
  
This Kotlin example performs the following tasks:  
  
1. Returns a list of the available DB engines.  
2. Creates a custom DB parameter group.  
3. Gets the parameter groups.  
4. Gets the parameters in the group.  
5. Modifies the auto_increment_increment parameter.  
6. Displays the updated parameter value.  
7. Gets a list of allowed engine versions.  
8. Creates an Aurora DB cluster database.  
9. Waits for DB instance to be ready.  
10. Gets a list of instance classes available for the selected engine.  
11. Creates a database instance in the cluster.  
12. Waits for the database instance in the cluster to be ready.  
13. Creates a snapshot.  
14. Waits for DB snapshot to be ready.  
15. Deletes the DB instance.  
16. Deletes the DB cluster.  
17. Deletes the DB cluster group.  
*/  
  
var slTime: Long = 20  
suspend fun main(args: Array<String>) {  
    val usage = """  
        Usage:  
        <dbClusterGroupName> <dbParameterGroupFamily> <dbInstanceClusterIdentifier>  
        <dbName> <dbSnapshotIdentifier> <username> <userPassword>  
  
        Where:  
        dbClusterGroupName - The database group name.  
        dbParameterGroupFamily - The database parameter group name.  
        dbInstanceClusterIdentifier - The database instance identifier.  
    """  
}
```

```
    dbName - The database name.
    dbSnapshotIdentifier - The snapshot identifier.
    username - The user name.
    userPassword - The password that corresponds to the user name.
"""

if (args.size != 7) {
    println(usage)
    exitProcess(1)
}

val dbClusterGroupName = args[0]
val dbParameterGroupFamily = args[1]
val dbInstanceClusterIdentifier = args[2]
val dbInstanceIdentifier = args[3]
val dbName = args[4]
val dbSnapshotIdentifier = args[5]
val username = args[6]
val userPassword = args[7]

println("1. Return a list of the available DB engines")
describeAuroraDBEngines()

println("2. Create a custom parameter group")
createDBClusterParameterGroup(dbClusterGroupName, dbParameterGroupFamily)

println("3. Get the parameter group")
describeDbClusterParameterGroups(dbClusterGroupName)

println("4. Get the parameters in the group")
describeDbClusterParameters(dbClusterGroupName, 0)

println("5. Modify the auto_increment_offset parameter")
modifyDBClusterParas(dbClusterGroupName)

println("6. Display the updated parameter value")
describeDbClusterParameters(dbClusterGroupName, -1)

println("7. Get a list of allowed engine versions")
getAllowedClusterEngines(dbParameterGroupFamily)

println("8. Create an Aurora DB cluster database")
val arnClusterVal = createDBCluster(dbClusterGroupName, dbName,
dbInstanceClusterIdentifier, username, userPassword)
println("The ARN of the cluster is $arnClusterVal")

println("9. Wait for DB instance to be ready")
waitForClusterInstanceReady(dbInstanceClusterIdentifier)

println("10. Get a list of instance classes available for the selected engine")
val instanceClass = getListInstanceClasses()

println("11. Create a database instance in the cluster.")
val clusterDBARN = createDBInstanceCluster(dbInstanceIdentifier,
dbInstanceClusterIdentifier, instanceClass)
println("The ARN of the database is $clusterDBARN")

println("12. Wait for DB instance to be ready")
waitForAuroraInstanceReady(dbInstanceIdentifier)

println("13. Create a snapshot")
createDBClusterSnapshot(dbInstanceClusterIdentifier, dbSnapshotIdentifier)

println("14. Wait for DB snapshot to be ready")
waitForSnapshotReady(dbSnapshotIdentifier, dbInstanceClusterIdentifier)
```

```
    println("15. Delete the DB instance")
    deleteDBInstance(dbInstanceIdentifier)

    println("16. Delete the DB cluster")
    deleteCluster(dbInstanceClusterIdentifier)

    println("17. Delete the DB cluster group")
    if (clusterDBARN != null) {
        deleteDBClusterGroup(dbClusterGroupName, clusterDBARN)
    }
    println("The Scenario has successfully completed.")
}

@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(dbClusterGroupName: String, clusterDBARN: String) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the database
                        ARN.
                            isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest = DeleteDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupName
        }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}

suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest = DeleteDbClusterRequest {
        dbClusterIdentifier = dbInstanceClusterIdentifier
        skipFinalSnapshot = true
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest = DeleteDbInstanceRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
        deleteAutomatedBackups = true
        skipFinalSnapshot = true
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is ${response.dbInstance?.dbInstanceState}")
    }
}

suspend fun waitSnapshotReady(dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest = DescribeDbClusterSnapshotsRequest {
        dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        dbClusterIdentifier = dbInstanceClusterIdentifier
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
        println("The Snapshot is available!")
    }
}

suspend fun createDBClusterSnapshot(dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?) {
    val snapshotRequest = CreateDbClusterSnapshotRequest {
        dbClusterIdentifier = dbInstanceClusterIdentifier
        dbClusterSnapshotIdentifier = dbSnapshotIdentifier
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}

suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest = DescribeDbInstancesRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
    }
}
```

```

var endpoint = ""
RdsClient { region = "us-west-2" }.use { rdsClient ->
    while (!instanceReady) {
        val response = rdsClient.describeDbInstances(instanceRequest)
        response.dbInstances?.forEach { instance ->
            instanceReadyStr = instance.dbInstanceStatus.toString()
            if (instanceReadyStr.contains("available")) {
                endpoint = instance.endpoint?.address.toString()
                instanceReady = true
            } else {
                print(".")
                delay(sleepTime * 1000)
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}

suspend fun createDBInstanceCluster(dbInstanceIdentifierVal: String?,
                                    dbInstanceClusterIdentifierVal: String?, instanceClassVal: String?): String? {
    val instanceRequest = CreateDbInstanceRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
        dbClusterIdentifier = dbInstanceClusterIdentifierVal
        engine = "aurora-mysql"
        dbInstanceClass = instanceClassVal
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceState}")
        return response.dbInstance?.dbInstanceArn
    }
}

suspend fun getListInstanceClasses(): String {
    val optionsRequest = DescribeOrderableDbInstanceOptionsRequest {
        engine = "aurora-mysql"
        maxRecords = 20
    }
    var instanceClass = ""
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
            instanceClass = instanceOption.dbInstanceClass.toString()
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
    return instanceClass
}

// Waits until the database instance is available.
suspend fun waitForClusterInstanceReady(dbClusterIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest = DescribeDbClustersRequest {
        dbClusterIdentifier = dbClusterIdentifierVal
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbClusters(instanceRequest)
            response.dbClusters?.forEach { cluster ->

```

```
        instanceReadyStr = cluster.status.toString()
        if (instanceReadyStr.contains("available")) {
            instanceReady = true
        } else {
            print(".")
            delay(sleepTime * 1000)
        }
    }
}
println("Database cluster is available!")
}

suspend fun createDBCluster(dbParameterGroupFamilyVal: String?, dbName: String?,
dbClusterIdentifierVal: String?, userName: String?, password: String?): String? {
    val clusterRequest = CreateDbClusterRequest {
        databaseName = dbName
        dbClusterIdentifier = dbClusterIdentifierVal
        dbClusterParameterGroupName = dbParameterGroupFamilyVal
        engine = "aurora-mysql"
        masterUsername = userName
        masterUserPassword = password
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest = DescribeDbEngineVersionsRequest {
        dbParameterGroupFamily = dbParameterGroupFamilyVal
        engine = "aurora-mysql"
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 = Parameter {
        parameterName = "auto_increment_offset"
        applyMethod = ApplyMethod.fromValue("immediate")
        parameterValue = "5"
    }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest = ModifyDbClusterParameterGroupRequest {
        dbClusterParameterGroupName = dClusterGroupName
        parameters = paraList
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
successfully modified")
    }
}
```

```

    }

    suspend fun describeDbClusterParameters(dbClusterGroupName: String?, flag: Int) {
        val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
        dbParameterGroupsRequest = if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbClusterGroupName
                source = "user"
            }
        }

        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
            response.parameters?.forEach { para ->
                // Only print out information about either auto_increment_offset or
                auto_increment_increment.
                val paraName = para.parameterName
                if (paraName != null) {
                    if (paraName.compareTo("auto_increment_offset") == 0 || paraName.compareTo("auto_increment_increment") == 0) {
                        println("**** The parameter name is $paraName")
                        println("**** The parameter value is ${para.parameterValue}")
                        println("**** The parameter data type is ${para.dataType}")
                        println("**** The parameter description is ${para.description}")
                        println("**** The parameter allowed values is
${para.allowedValues}")
                    }
                }
            }
        }
    }

    suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
        val groupsRequest = DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
            response.dbClusterParameterGroups?.forEach { group ->
                println("The group name is ${group.dbClusterParameterGroupName}")
                println("The group ARN is ${group.dbClusterParameterGroupArn}")
            }
        }
    }

    suspend fun createDBClusterParameterGroup(dbClusterGroupNameVal: String?,
                                              dbParameterGroupFamilyVal: String?) {
        val groupRequest = CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.createDbClusterParameterGroup(groupRequest)
            println("The group name is
${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
        }
    }
}

```

```
suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest = DescribeDbEngineVersionsRequest {
        engine = "aurora-mysql"
        defaultOnly = true
        maxRecords = 20
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engine0b ->
            println("The name of the DB parameter group family for the database engine
is ${engine0b.dbParameterGroupFamily}")
            println("The name of the database engine ${engine0b.engine}")
            println("The version number of the database engine
${engine0b.engineVersion}")
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.

- [CreateDBCluster](#)
- [CreateDBClusterParameterGroup](#)
- [CreateDBClusterSnapshot](#)
- [CreateDBInstance](#)
- [DeleteDBCluster](#)
- [DeleteDBClusterParameterGroup](#)
- [DeleteDBInstance](#)
- [DescribeDBClusterParameterGroups](#)
- [DescribeDBClusterParameters](#)
- [DescribeDBClusterSnapshots](#)
- [DescribeDBClusters](#)
- [DescribeDBEngineVersions](#)
- [DescribeDBInstances](#)
- [DescribeOrderableDBInstanceOptions](#)
- [ModifyDBClusterParameterGroup](#)

## CloudWatch examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon CloudWatch.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3516\)](#)

## Actions

### Create an alarm that watches a metric

The following code example shows how to create an Amazon CloudWatch alarm that watches a metric.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun putMetricAlarm(alarmNameVal: String, instanceIdVal: String) {

    val dimension0b = Dimension {
        name = "InstanceId"
        value = instanceIdVal
    }

    val request = PutMetricAlarmRequest {
        alarmName = alarmNameVal
        comparisonOperator = ComparisonOperator.GreaterThanThreshold
        evaluationPeriods = 1
        metricName = "CPUUtilization"
        namespace = "AWS/EC2"
        period = 60
        statistic = Statistic.fromValue("Average")
        threshold = 70.0
        actionsEnabled = false
        alarmDescription = "An Alarm created by the Kotlin SDK when server CPU
utilization exceeds 70%"
        unit = StandardUnit.fromValue("Seconds")
        dimensions = listOf(dimension0b)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
}
```

- For API details, see [PutMetricAlarm in AWS SDK for Kotlin API reference](#).

## Delete alarms

The following code example shows how to delete Amazon CloudWatch alarms.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteCWAAlarm(alarmNameVal: String) {

    val request = DeleteAlarmsRequest {
        alarmNames = listOf(alarmNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
    }
}
```

```
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- For API details, see [DeleteAlarms](#) in *AWS SDK for Kotlin API reference*.

## Describe alarms for a metric

The following code example shows how to describe Amazon CloudWatch alarms for a metric.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun desCWAAlarms() {

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(DescribeAlarmsRequest {})
        response.metricAlarms?.forEach { alarm ->
            println("Retrieved alarm ${alarm.alarmName}")
        }
    }
}
```

- For API details, see [DescribeAlarmsForMetric](#) in *AWS SDK for Kotlin API reference*.

## Disable alarm actions

The following code example shows how to disable Amazon CloudWatch alarm actions.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun disableActions(alarmName: String) {

    val request = DisableAlarmActionsRequest {
        alarmNames = listOf(alarmName)
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
}
```

- For API details, see [DisableAlarmActions](#) in *AWS SDK for Kotlin API reference*.

## Enable alarm actions

The following code example shows how to enable Amazon CloudWatch alarm actions.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun enableActions(alarm: String) {  
  
    val request = EnableAlarmActionsRequest {  
        alarmNames = listOf(alarm)  
    }  
  
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->  
        cwClient.enableAlarmActions(request)  
        println("Successfully enabled actions on alarm $alarm")  
    }  
}
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for Kotlin API reference*.

## CloudWatch Logs examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon CloudWatch Logs.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3519\)](#)

## Actions

### Delete a subscription filter

The following code example shows how to delete an Amazon CloudWatch Logs subscription filter.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteSubFilter(filter: String?, logGroup: String?) {  
  
    val request = DeleteSubscriptionFilterRequest {
```

```
        filterName = filter
        logGroupName = logGroup
    }

    CloudWatchLogsClient { region = "us-west-2" }.use { logs ->
        logs.deleteSubscriptionFilter(request)
        println("Successfully deleted CloudWatch logs subscription filter named
$filter")
    }
}
```

- For API details, see [DeleteSubscriptionFilter](#) in *AWS SDK for Kotlin API reference*.

## Describe existing subscription filters

The following code example shows how to describe Amazon CloudWatch Logs existing subscription filters.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeFilters(logGroup: String) {

    val request = DescribeSubscriptionFiltersRequest {
        logGroupName = logGroup
        limit = 1
    }

    CloudWatchLogsClient { region = "us-west-2" }.use { cwlClient ->
        val response = cwlClient.describeSubscriptionFilters(request)
        response.subscriptionFilters?.forEach { filter ->
            println("Retrieved filter with name ${filter.filterName} pattern
${filter.filterPattern} and destination ${filter.destinationArn}")
        }
    }
}
```

- For API details, see [DescribeSubscriptionFilters](#) in *AWS SDK for Kotlin API reference*.

## Amazon Cognito Identity Provider examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Cognito Identity Provider.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3521\)](#)
- [Scenarios \(p. 3526\)](#)

## Actions

### Confirm a user

The following code example shows how to confirm an Amazon Cognito user.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun confirmSignUp(clientIdVal: String?, codeVal: String?, userNameVal: String?) {
    val signUpRequest = ConfirmSignUpRequest {
        clientId = clientIdVal
        confirmationCode = codeVal
        username = userNameVal
    }
    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal was confirmed")
    }
}
```

- For API details, see [ConfirmSignUp in AWS SDK for Kotlin API reference](#).

### Get a token to associate an MFA application with a user

The following code example shows how to get a token to associate an MFA application with an Amazon Cognito user.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest = AssociateSoftwareTokenRequest {
        session = sessionVal
    }
    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient ->
```

```
    val tokenResponse =
identityProviderClient.associateSoftwareToken(softwareTokenRequest)
    val secretCode = tokenResponse.secretCode
    println("Enter this token into Google Authenticator")
    println(secretCode)
    return tokenResponse.session
}
}
```

- For API details, see [AssociateSoftwareToken](#) in *AWS SDK for Kotlin API reference*.

## Get information about a user

The following code example shows how to get information about an Amazon Cognito user.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getAdminUser(userNameVal: String?, poolIdVal: String?) {

    val userRequest = Admin GetUserRequest {
        username = userNameVal
        userPoolId = poolIdVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
    ->
        val response = identityProviderClient.admin GetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

- For API details, see [Admin GetUser](#) in *AWS SDK for Kotlin API reference*.

## List users

The following code example shows how to list Amazon Cognito users.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAllUsers(userPoolId: String) {

    val request = ListUsersRequest {
        this.userPoolId = userPoolId
    }
}
```

```
CognitoIdentityProviderClient { region = "us-east-1" }.use { cognitoClient ->
    val response = cognitoClient.listUsers(request)
    response.users?.forEach { user ->
        println("The user name is ${user.username}")
    }
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Kotlin API reference*.

## Resend a confirmation code

The following code example shows how to resend an Amazon Cognito confirmation code.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun resendConfirmationCode(clientIdVal: String?, userNameVal: String?) {

    val codeRequest = ResendConfirmationCodeRequest {
        clientId = clientIdVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
            (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- For API details, see [ResendConfirmationCode](#) in *AWS SDK for Kotlin API reference*.

## Respond to an authentication challenge

The following code example shows how to respond to an Amazon Cognito authentication challenge.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(userName: String, clientIdVal: String?,
    mfaCode: String, sessionVal: String?) {

    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponses0b = mutableMapOf<String, String>()
```

```
challengeResponses0b["USERNAME"] = userName
challengeResponses0b["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

val respondToAuthChallengeRequest = RespondToAuthChallengeRequest {
    challengeName = ChallengeNameType.SoftwareTokenMfa
    clientId = clientIdVal
    challengeResponses = challengeResponses0b
    session = sessionVal
}

CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
->
    val respondToAuthChallengeResult =
        identityProviderClient.respondToAuthChallenge(respondToAuthChallengeRequest)
    println("respondToAuthChallengeResult.getAuthenticationResult()")
    ${respondToAuthChallengeResult.authenticationResult}
}
}
```

- For API details, see [AdminRespondToAuthChallenge](#) in *AWS SDK for Kotlin API reference*.

## Sign up a user

The following code example shows how to sign up a user with Amazon Cognito.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun signUp(clientIdVal: String?, userNameVal: String?, passwordVal: String?,
emailVal: String?) {

    val userAttrs = AttributeType {
        name = "email"
        value = emailVal
    }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)

    val signUpRequest = SignUpRequest {
        userAttributes = userAttrsList
        username = userNameVal
        clientId = clientIdVal
        password = passwordVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- For API details, see [SignUp](#) in *AWS SDK for Kotlin API reference*.

## Start authentication with administrator credentials

The following code example shows how to start authentication with Amazon Cognito and administrator credentials.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun initiateAuth(clientIdVal: String?, userNameVal: String, passwordVal: String): InitiateAuthResponse {  
  
    val authParas = mutableMapOf <String, String>()  
    authParas["USERNAME"] = userNameVal  
    authParas["PASSWORD"] = passwordVal  
  
    val authRequest = InitiateAuthRequest {  
        clientId = clientIdVal  
        authParameters = authParas  
        authFlow = AuthFlowType.UserPasswordAuth  
    }  
  
    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient ->  
        val response = identityProviderClient.initiateAuth(authRequest)  
        println("Result Challenge is ${response.challengeName}")  
        return response  
    }  
}
```

- For API details, see [AdminInitiateAuth](#) in *AWS SDK for Kotlin API reference*.

## Verify an MFA application with a user

The following code example shows how to verify an MFA application with an Amazon Cognito user.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Verify the TOTP and register for MFA.  
suspend fun verifyTOTP(sessionVal: String?, codeVal: String?) {  
  
    val tokenRequest = VerifySoftwareTokenRequest {  
        userCode = codeVal  
        session = sessionVal  
    }  
  
    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient ->
```

```
        val verifyResponse = identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
```

- For API details, see [VerifySoftwareToken](#) in *AWS SDK for Kotlin API reference*.

## Scenarios

### Sign up a user with a user pool that requires MFA

The following code example shows how to:

- Sign up a user with a user name, password, and email address.
- Confirm the user from a code sent in email.
- Set up multi-factor authentication by associating an MFA application with the user.
- Sign in by using a password and an MFA code.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

TIP: To set up the required user pool, run the AWS Cloud Development
Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
cognito_scenario_user_pool_with_mfa.

This code example performs the following operations:

1. Invokes the signUp method to sign up a user.
2. Invokes the adminGetUser method to get the user's confirmation status.
3. Invokes the ResendConfirmationCode method if the user requested another code.
4. Invokes the confirmSignUp method.
5. Invokes the initiateAuth to sign in. This results in being prompted to set up TOTP
(time-based one-time password). (The response is "ChallengeName": "MFA_SETUP").
6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key. This
can be used with Google Authenticator.
7. Invokes the VerifySoftwareToken method to verify the TOTP and register for MFA.
8. Invokes the AdminInitiateAuth to sign in again. This results in being prompted to
submit a TOTP (Response: "ChallengeName": "SOFTWARE_TOKEN_MFA").
9. Invokes the AdminRespondToAuthChallenge to get back a token.
*/
```

```
suspend fun main(args: Array<String>) {

    val usage = """
        Usage:
            <clientId> <poolId>
```

```

    Where:
        clientId - The app client Id value that you can get from the AWS CDK
script.
        poolId - The pool Id that you can get from the AWS CDK script.
    """

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    val clientId = args[0]
    val poolId = args[1]

    // Use the console to get data from the user.
    println("/** Enter your use name")
    val in0b = Scanner(System.`in`)
    val userName = in0b.nextLine()
    println(userName)

    println("/** Enter your password")
    val password: String = in0b.nextLine()

    println("/** Enter your email")
    val email = in0b.nextLine()

    println("/** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("/** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("/** Conformation code sent to $userName. Would you like to send a new
code? (Yes/No)")
    val ans = in0b.nextLine()

    if (ans.compareTo("Yes") == 0) {
        println("/** Sending a new confirmation code")
        resendConfirmationCode(clientId, userName)
    }
    println("/** Enter the confirmation code that was emailed")
    val code = in0b.nextLine()
    confirmSignUp(clientId, code, userName)

    println("/** Rechecking the status of $userName in the user pool")
    getAdminUser(userName, poolId)

    val authResponse = initiateAuth(clientId, userName, password)
    val mySession = authResponse.session
    val newSession = getSecretForAppMFA(mySession)
    println("/** Enter the 6-digit code displayed in Google Authenticator")
    val myCode = in0b.nextLine()

    // Verify the TOTP and register for MFA.
    verifyTOTP(newSession, myCode)
    println("/** Re-enter a 6-digit code displayed in Google Authenticator")
    val mfaCode: String = in0b.nextLine()
    val authResponse1 = initiateAuth(clientId, userName, password)
    val session2 = authResponse1.session
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun resendConfirmationCode(clientIdVal: String?, userNameVal: String?) {

    val codeRequest = ResendConfirmationCodeRequest {
        clientId = clientIdVal
}

```

```

        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
->
    val response = identityProviderClient.resendConfirmationCode(codeRequest)
    println("Method of delivery is " +
(response.codeDeliveryDetails?.deliveryMedium))
    }
}

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(userName: String, clientIdVal: String?,
mfaCode: String, sessionVal: String?) {

    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponses0b = mutableMapOf<String, String>()
    challengeResponses0b["USERNAME"] = userName
    challengeResponses0b["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val respondToAuthChallengeRequest = RespondToAuthChallengeRequest {
        challengeName = ChallengeNameType.SoftwareTokenMfa
        clientId = clientIdVal
        challengeResponses = challengeResponses0b
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
->
    val respondToAuthChallengeResult =
identityProviderClient.respondToAuthChallenge(respondToAuthChallengeRequest)
    println("respondToAuthChallengeResult.getAuthenticationResult() ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(sessionVal: String?, codeVal: String?) {

    val tokenRequest = VerifySoftwareTokenRequest {
        userCode = codeVal
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
->
    val verifyResponse = identityProviderClient.verifySoftwareToken(tokenRequest)
    println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {

    val softwareTokenRequest = AssociateSoftwareTokenRequest {
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
->
    val tokenResponse =
identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}

```

```
}

suspend fun initiateAuth(clientIdVal: String?, userNameVal: String, passwordVal: String): InitiateAuthResponse {

    val authParas = mutableMapOf <String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest = InitiateAuthRequest {
        clientId = clientIdVal
        authParameters = authParas
        authFlow = AuthFlowType.UserPasswordAuth
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
    ->
        val response = identityProviderClient.initiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}

suspend fun confirmSignUp(clientIdVal: String?, codeVal: String?, userNameVal: String?) {

    val signUpRequest = ConfirmSignUpRequest {
        clientId = clientIdVal
        confirmationCode = codeVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
    ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal was confirmed")
    }
}

suspend fun getAdminUser(userNameVal: String?, poolIdVal: String?) {

    val userRequest = Admin GetUserRequest {
        username = userNameVal
        userPoolId = poolIdVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
    ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}

suspend fun signUp(clientIdVal: String?, userNameVal: String?, passwordVal: String?, emailVal: String?) {

    val userAttrs = AttributeType {
        name = "email"
        value = emailVal
    }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)

    val signUpRequest = SignUpRequest {
        userAttributes = userAttrsList
    }
}
```

```
        username = userNameVal
        clientId = clientIdVal
        password = passwordVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { identityProviderClient
->
    identityProviderClient.signUp(signUpRequest)
    println("User has been signed up")
}
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - [Admin GetUser](#)
  - [Admin Initiate Auth](#)
  - [Admin Respond To Auth Challenge](#)
  - [Associate Software Token](#)
  - [Confirm Device](#)
  - [Confirm Sign Up](#)
  - [Initiate Auth](#)
  - [List Users](#)
  - [Resend Confirmation Code](#)
  - [Respond To Auth Challenge](#)
  - [Sign Up](#)
  - [Verify Software Token](#)

## DynamoDB examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon DynamoDB.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3530\)](#)
- [Scenarios \(p. 3536\)](#)

## Actions

### Create a table

The following code example shows how to create a DynamoDB table.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createNewTable(tableNameVal: String, key: String): String? {  
  
    val attDef = AttributeDefinition {  
        attributeName = key  
        attributeType = ScalarAttributeType.S  
    }  
  
    val keySchemaVal = KeySchemaElement {  
        attributeName = key  
        keyType = KeyType.Hash  
    }  
  
    val provisionedVal = ProvisionedThroughput {  
        readCapacityUnits = 10  
        writeCapacityUnits = 10  
    }  
  
    val request = CreateTableRequest {  
        attributeDefinitions = listOf(attDef)  
        keySchema = listOf(keySchemaVal)  
        provisionedThroughput = provisionedVal  
        tableName = tableNameVal  
    }  
  
    DynamoDbClient { region = "us-east-1" }.use { ddb ->  
  
        var tableArn: String  
        val response = ddb.createTable(request)  
        ddb.waitUntilTableExists { // suspend call  
            tableName = tableNameVal  
        }  
        tableArn = response.tableDescription!!.tableArn.toString()  
        println("Table $tableArn is ready")  
        return tableArn  
    }  
}
```

- For API details, see [CreateTable](#) in *AWS SDK for Kotlin API reference*.

## Delete a table

The following code example shows how to delete a DynamoDB table.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteDynamoDBTable(tableNameVal: String) {  
  
    val request = DeleteTableRequest {  
        tableName = tableNameVal  
    }  
  
    DynamoDbClient { region = "us-east-1" }.use { ddb ->  
        ddb.deleteTable(request)  
        println("$tableNameVal was deleted")  
    }  
}
```

```
    }
```

- For API details, see [DeleteTable](#) in *AWS SDK for Kotlin API reference*.

## Delete an item from a table

The following code example shows how to delete an item from a DynamoDB table.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteDymamoDBItem(tableNameVal: String, keyName: String, keyVal: String) {  
  
    val keyToGet = mutableMapOf<String, AttributeValue>()  
    keyToGet[keyName] = AttributeValue.S(keyVal)  
  
    val request = DeleteItemRequest {  
        tableName = tableNameVal  
        key = keyToGet  
    }  
  
    DynamoDbClient { region = "us-east-1" }.use { ddb ->  
        ddb.deleteItem(request)  
        println("Item with key matching $keyVal was deleted")  
    }  
}
```

- For API details, see [DeleteItem](#) in *AWS SDK for Kotlin API reference*.

## Get an item from a table

The following code example shows how to get an item from a DynamoDB table.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getSpecificItem(tableNameVal: String, keyName: String, keyVal: String) {  
  
    val keyToGet = mutableMapOf<String, AttributeValue>()  
    keyToGet[keyName] = AttributeValue.S(keyVal)  
  
    val request = GetItemRequest {  
        key = keyToGet  
        tableName = tableNameVal  
    }  
}
```

```
DynamoDbClient { region = "us-east-1" }.use { ddb ->
    val returnedItem = ddb.getItem(request)
    val numbersMap = returnedItem.item
    numbersMap?.forEach { key1 ->
        println(key1.key)
        println(key1.value)
    }
}
```

- For API details, see [GetItem](#) in *AWS SDK for Kotlin API reference*.

## List tables

The following code example shows how to list DynamoDB tables.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAllTables() {

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.listTables(ListTablesRequest {})
        response.tableNames?.forEach { tableName ->
            println("Table name is $tableName")
        }
    }
}
```

- For API details, see [ListTables](#) in *AWS SDK for Kotlin API reference*.

## Put an item in a table

The following code example shows how to put an item in a DynamoDB table.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun putItemInTable(
    tableNameVal: String,
    key: String,
    keyVal: String,
    albumTitle: String,
    albumTitleValue: String,
    awards: String,
```

```
        awardVal: String,
        songTitle: String,
        songTitleVal: String
    ) {
    val itemValues = mutableMapOf<String, AttributeValue>()

    // Add all content to the table.
    itemValues[key] = AttributeValue.S(keyVal)
    itemValues[songTitle] = AttributeValue.S(songTitleVal)
    itemValues[albumTitle] = AttributeValue.S(albumTitleValue)
    itemValues[awards] = AttributeValue.S(awardVal)

    val request = PutItemRequest {
        tableName = tableNameVal
        item = itemValues
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println(" A new item was placed into $tableNameVal.")
    }
}
```

- For API details, see [PutItem](#) in *AWS SDK for Kotlin API reference*.

## Query a table

The following code example shows how to query a DynamoDB table.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun queryDynTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionKeyVal: String,
    partitionAlias: String
): Int {

    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = partitionKeyName

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.S(partitionKeyVal)

    val request = QueryRequest {
        tableName = tableNameVal
        keyConditionExpression = "$partitionAlias = :$partitionKeyName"
        expressionAttributeNames = attrNameAlias
        this.expressionAttributeValues = attrValues
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}
```

```
    }
```

- For API details, see [Query](#) in *AWS SDK for Kotlin API reference*.

## Scan a table

The following code example shows how to scan a DynamoDB table.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun scanItems(tableNameVal: String) {  
  
    val request = ScanRequest {  
        tableName = tableNameVal  
    }  
  
    DynamoDbClient { region = "us-east-1" }.use { ddb ->  
        val response = ddb.scan(request)  
        response.items?.forEach { item ->  
            item.keys.forEach { key ->  
                println("The key name is $key\n")  
                println("The value is ${item[key]}")  
            }  
        }  
    }  
}
```

- For API details, see [Scan](#) in *AWS SDK for Kotlin API reference*.

## Update an item in a table

The following code example shows how to update an item in a DynamoDB table.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun updateTableItem(  
    tableNameVal: String,  
    keyName: String,  
    keyVal: String,  
    name: String,  
    updateVal: String  
) {  
  
    val itemKey = mutableMapOf<String, AttributeValue>()
```

```
    itemKey[keyName] = AttributeValue.S(keyVal)

    val updatedValues = mutableMapOf<String, AttributeValueUpdate>()
    updatedValues[name] = AttributeValueUpdate {
        value = AttributeValue.S(updateVal)
        action = AttributeAction.Put
    }

    val request = UpdateItemRequest {
        tableName = tableNameVal
        key = itemKey
        attributeUpdates = updatedValues
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.updateItem(request)
        println("Item in $tableNameVal was updated")
    }
}
```

- For API details, see [UpdateItem](#) in *AWS SDK for Kotlin API reference*.

## Scenarios

### Get started using tables, items, and queries

The following code example shows how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.
- Delete a movie from the table.
- Delete the table.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a DynamoDB table.

```
suspend fun createScenarioTable(tableNameVal: String, key: String) {

    val attDef = AttributeDefinition {
        attributeName = key
        attributeType = ScalarAttributeType.N
    }

    val attDef1 = AttributeDefinition {
        attributeName = "title"
        attributeType = ScalarAttributeType.S
    }
```

```

val keySchemaVal = KeySchemaElement {
    attributeName = key
    keyType = KeyType.Hash
}

val keySchemaVal1 = KeySchemaElement {
    attributeName = "title"
    keyType = KeyType.Range
}

val provisionedVal = ProvisionedThroughput {
    readCapacityUnits = 10
    writeCapacityUnits = 10
}

val request = CreateTableRequest {
    attributeDefinitions = listOf(attDef, attDef1)
    keySchema = listOf(keySchemaVal, keySchemaVal1)
    provisionedThroughput = provisionedVal
    tableName = tableNameVal
}

DynamoDbClient { region = "us-east-1" }.use { ddb ->

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists { // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
    ${response.tableDescription?.tableArn}")
}
}

```

Create a helper function to download and extract the sample JSON file.

```

// Load data into the table.
suspend fun loadData(tableName: String, fileName: String) {

    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {

        if (t == 50)
            break

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()
        putMovie(tableName, year, title, info)
        t++
    }
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String
) {
}

```

```
val itemValues = mutableMapOf<String, AttributeValue>()
val strVal = year.toString()
// Add all content to the table.
itemValues["year"] = AttributeValue.N(strVal)
itemValues["title"] = AttributeValue.S(title)
itemValues["info"] = AttributeValue.S(info)

val request = PutItemRequest {
    tableName = tableNameVal
    item = itemValues
}

DynamoDbClient { region = "us-east-1" }.use { ddb ->
    ddb.putItem(request)
    println("Added $title to the Movie table.")
}
}
```

Get an item from a table.

```
suspend fun getMovie(tableNameVal: String, keyName: String, keyVal: String) {

    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request = GetItemRequest {
        key = keyToGet
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}
```

Full example.

```
suspend fun main(args: Array<String>) {

    val usage = """
        Usage:
        <fileName>

        Where:
        fileName - The path to the moviedata.json you can download from the Amazon
        DynamoDB Developer Guide.
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(1)
    }

    // Get the moviedata.json from the Amazon DynamoDB Developer Guide.
    val tableName = "Movies"
    val fileName = args[0]
```

```
    val partitionAlias = "#a"

    println("Creating an Amazon DynamoDB table named Movies with a key named id and a
sort key named title.")
    createScenarioTable(tableName, "year")
    loadData(tableName, fileName)
    getMovie(tableName, "year", "1933")
    scanMovies(tableName)
    val count = queryMovieTable(tableName, "year", partitionAlias)
    println("There are $count Movies released in 2013.")
    deleteIssuesTable(tableName)
}

suspend fun createScenarioTable(tableNameVal: String, key: String) {

    val attDef = AttributeDefinition {
        attributeName = key
        attributeType = ScalarAttributeType.N
    }

    val attDef1 = AttributeDefinition {
        attributeName = "title"
        attributeType = ScalarAttributeType.S
    }

    val keySchemaVal = KeySchemaElement {
        attributeName = key
        keyType = KeyType.Hash
    }

    val keySchemaVal1 = KeySchemaElement {
        attributeName = "title"
        keyType = KeyType.Range
    }

    val provisionedVal = ProvisionedThroughput {
        readCapacityUnits = 10
        writeCapacityUnits = 10
    }

    val request = CreateTableRequest {
        attributeDefinitions = listOf(attDef, attDef1)
        keySchema = listOf(keySchemaVal, keySchemaVal1)
        provisionedThroughput = provisionedVal
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->

        val response = ddb.createTable(request)
        ddb.waitUntilTableExists { // suspend call
            tableName = tableNameVal
        }
        println("The table was successfully created
${response.tableDescription?.tableArn}")
    }
}

// Load data into the table.
suspend fun loadData(tableName: String, fileName: String) {

    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
```

```
var t = 0
while (iter.hasNext()) {

    if (t == 50)
        break

    currentNode = iter.next() as ObjectNode
    val year = currentNode.path("year").asInt()
    val title = currentNode.path("title").asText()
    val info = currentNode.path("info").toString()
    putMovie(tableName, year, title, info)
    t++
}
}

suspend fun putMovie(
    tableNameVal: String,
    year: Int,
    title: String,
    info: String
) {
    val itemValues = mutableMapOf<String, AttributeValue>()
    val strVal = year.toString()
    // Add all content to the table.
    itemValues["year"] = AttributeValue.N(strVal)
    itemValues["title"] = AttributeValue.S(title)
    itemValues["info"] = AttributeValue.S(info)

    val request = PutItemRequest {
        tableName = tableNameVal
        item = itemValues
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.putItem(request)
        println("Added $title to the Movie table.")
    }
}

suspend fun getMovie(tableNameVal: String, keyName: String, keyVal: String) {

    val keyToGet = mutableMapOf<String, AttributeValue>()
    keyToGet[keyName] = AttributeValue.N(keyVal)
    keyToGet["title"] = AttributeValue.S("King Kong")

    val request = GetItemRequest {
        key = keyToGet
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val returnedItem = ddb.getItem(request)
        val numbersMap = returnedItem.item
        numbersMap?.forEach { key1 ->
            println(key1.key)
            println(key1.value)
        }
    }
}

suspend fun deleteIssuesTable(tableNameVal: String) {

    val request = DeleteTableRequest {
        tableName = tableNameVal
    }
}
```

```

        DynamoDbClient { region = "us-east-1" }.use { ddb ->
            ddb.deleteTable(request)
            println("$tableNameVal was deleted")
        }
    }

suspend fun queryMovieTable(
    tableNameVal: String,
    partitionKeyName: String,
    partitionAlias: String
): Int {

    val attrNameAlias = mutableMapOf<String, String>()
    attrNameAlias[partitionAlias] = "year"

    // Set up mapping of the partition name with the value.
    val attrValues = mutableMapOf<String, AttributeValue>()
    attrValues[":$partitionKeyName"] = AttributeValue.N("2013")

    val request = QueryRequest {
        tableName = tableNameVal
        keyConditionExpression = "$partitionAlias = :$partitionKeyName"
        expressionAttributeNames = attrNameAlias
        this.expressionAttributeValues = attrValues
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.query(request)
        return response.count
    }
}

suspend fun scanMovies(tableNameVal: String) {

    val request = ScanRequest {
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        val response = ddb.scan(request)
        response.items?.forEach { item ->
            item.keys.forEach { key ->
                println("The key name is $key\n")
                println("The value is ${item[key]}")
            }
        }
    }
}

```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.

- [BatchWriteItem](#)
- [CreateTable](#)
- [DeleteItem](#)
- [DeleteTable](#)
- [DescribeTable](#)
- [GetItem](#)
- [PutItem](#)
- [Query](#)
- [Scan](#)
- [UpdateItem](#)

## Query a table by using batches of PartiQL statements

The following code example shows how to:

- Get a batch of items by running multiple SELECT statements.
- Add a batch of items by running multiple INSERT statements.
- Update a batch of items by running multiple UPDATE statements.
- Delete a batch of items by running multiple DELETE statements.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun main() {  
  
    val ddb = DynamoDbClient { region = "us-east-1" }  
    val tableName = "MoviesPartiQLBatch"  
    println("Creating an Amazon DynamoDB table named $tableName with a key named id and  
    a sort key named title.")  
    createTablePartiQLBatch(ddb, tableName, "year")  
    putRecordBatch(ddb)  
    updateTableItemBatchBatch(ddb)  
    deleteItemsBatch(ddb)  
    deleteTablePartiQLBatch(tableName)  
}  
  
suspend fun createTablePartiQLBatch(ddb: DynamoDbClient, tableNameVal: String, key:  
String) {  
  
    val attDef = AttributeDefinition {  
        attributeName = key  
        attributeType = ScalarAttributeType.N  
    }  
  
    val attDef1 = AttributeDefinition {  
        attributeName = "title"  
        attributeType = ScalarAttributeType.S  
    }  
  
    val keySchemaVal = KeySchemaElement {  
        attributeName = key  
        keyType = KeyType.Hash  
    }  
  
    val keySchemaVal1 = KeySchemaElement {  
        attributeName = "title"  
        keyType = KeyType.Range  
    }  
  
    val provisionedVal = ProvisionedThroughput {  
        readCapacityUnits = 10  
        writeCapacityUnits = 10  
    }  
  
    val request = CreateTableRequest {  
        attributeDefinitions = listOf(attDef, attDef1)  
        keySchema = listOf(keySchemaVal, keySchemaVal1)  
    }  
}
```

```
        provisionedThroughput = provisionedVal
        tableName = tableNameVal
    }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists { // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
    ${response.tableDescription?.tableArn}")
}

suspend fun putRecordBatch(ddb: DynamoDbClient) {

    val sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?, 'title' : ?, 'info' : ?}"

    // Create three movies to add to the Amazon DynamoDB table.
    val parametersMovie1 = mutableListOf<AttributeValue>()
    parametersMovie1.add(AttributeValue.N("2022"))
    parametersMovie1.add(AttributeValue.S("My Movie 1"))
    parametersMovie1.add(AttributeValue.S("No Information"))

    val statementRequestMovie1 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersMovie1
    }

    // Set data for Movie 2.
    val parametersMovie2 = mutableListOf<AttributeValue>()
    parametersMovie2.add(AttributeValue.N("2022"))
    parametersMovie2.add(AttributeValue.S("My Movie 2"))
    parametersMovie2.add(AttributeValue.S("No Information"))

    val statementRequestMovie2 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersMovie2
    }

    // Set data for Movie 3.
    val parametersMovie3 = mutableListOf<AttributeValue>()
    parametersMovie3.add(AttributeValue.N("2022"))
    parametersMovie3.add(AttributeValue.S("My Movie 3"))
    parametersMovie3.add(AttributeValue.S("No Information"))

    val statementRequestMovie3 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersMovie3
    }

    // Add all three movies to the list.
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestMovie1)
    myBatchStatementList.add(statementRequestMovie2)
    myBatchStatementList.add(statementRequestMovie3)

    val batchRequest = BatchExecuteStatementRequest {
        statements = myBatchStatementList
    }
    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: " + response.toString())
    println("Added new movies using a batch command.")
}

suspend fun updateTableItemBatchBatch(ddb: DynamoDbClient) {
    val sqlStatement =
```

```
"UPDATE MoviesPartiQBatch SET info = 'directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack' where year=? and title=?"
```

```
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))
    val statementRequestRec1 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec1
    }
```

```
// Update record 2.
```

```
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec2
    }
```

```
// Update record 3.
```

```
    val parametersRec3 = mutableListOf<AttributeValue>()
    parametersRec3.add(AttributeValue.N("2022"))
    parametersRec3.add(AttributeValue.S("My Movie 3"))
    val statementRequestRec3 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec3
    }
```

```
// Add all three movies to the list.
```

```
    val myBatchStatementList = mutableListOf<BatchStatementRequest>()
    myBatchStatementList.add(statementRequestRec1)
    myBatchStatementList.add(statementRequestRec2)
    myBatchStatementList.add(statementRequestRec3)
```

```
    val batchRequest = BatchExecuteStatementRequest {
        statements = myBatchStatementList
    }
```

```
    val response = ddb.batchExecuteStatement(batchRequest)
    println("ExecuteStatement successful: $response")
    println("Updated three movies using a batch command.")
    println("Items were updated!")
}
```

```
suspend fun deleteItemsBatch(ddb: DynamoDbClient) {
```

```
// Specify three records to delete.
```

```
    val sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and title=?"
```

```
    val parametersRec1 = mutableListOf<AttributeValue>()
    parametersRec1.add(AttributeValue.N("2022"))
    parametersRec1.add(AttributeValue.S("My Movie 1"))
```

```
    val statementRequestRec1 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec1
    }
```

```
// Specify record 2.
```

```
    val parametersRec2 = mutableListOf<AttributeValue>()
    parametersRec2.add(AttributeValue.N("2022"))
    parametersRec2.add(AttributeValue.S("My Movie 2"))
    val statementRequestRec2 = BatchStatementRequest {
        statement = sqlStatement
        parameters = parametersRec2
    }
```

```
// Specify record 3.
val parametersRec3 = mutableListOf<AttributeValue>()
parametersRec3.add(AttributeValue.N("2022"))
parametersRec3.add(AttributeValue.S("My Movie 3"))
val statementRequestRec3 = BatchStatementRequest {
    statement = sqlStatement
    parameters = parametersRec3
}

// Add all three movies to the list.
val myBatchStatementList = mutableListOf<BatchStatementRequest>()
myBatchStatementList.add(statementRequestRec1)
myBatchStatementList.add(statementRequestRec2)
myBatchStatementList.add(statementRequestRec3)

val batchRequest = BatchExecuteStatementRequest {
    statements = myBatchStatementList
}

ddb.batchExecuteStatement(batchRequest)
println("Deleted three movies using a batch command.")
}

suspend fun deleteTablePartiQLBatch(tableNameVal: String) {

    val request = DeleteTableRequest {
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Kotlin API reference*.

## Query a table using PartiQL

The following code example shows how to:

- Get an item by running a SELECT statement.
- Add an item by running an INSERT statement.
- Update an item by running an UPDATE statement.
- Delete an item by running a DELETE statement.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun main(args: Array<String>) {

    val usage = """
        Usage:
        <fileName>
```

```
Where:  
    fileName - The path to the moviedata.json you can download from the Amazon  
DynamoDB Developer Guide.  
    """  
  
    if (args.size != 1) {  
        println(usage)  
        exitProcess(1)  
    }  
  
    val ddb = DynamoDbClient { region = "us-east-1" }  
    val tableName = "MoviesPartiQ"  
  
    // Get the moviedata.json from the Amazon DynamoDB Developer Guide.  
    val fileName = args[0]  
    println("Creating an Amazon DynamoDB table named MoviesPartiQ with a key named id  
and a sort key named title.")  
    createTablePartiQL(ddb, tableName, "year")  
    loadDataPartiQL(ddb, fileName)  
  
    println("***** Getting data from the MoviesPartiQ table.")  
    getMoviePartiQL(ddb)  
  
    println("***** Putting a record into the MoviesPartiQ table.")  
    putRecordPartiQL(ddb)  
  
    println("***** Updating a record.")  
    updateTableItemPartiQL(ddb)  
  
    println("***** Querying the movies released in 2013.")  
    queryTablePartiQL(ddb)  
  
    println("***** Deleting the MoviesPartiQ table.")  
    deleteTablePartiQL(tableName)  
}  
  
suspend fun createTablePartiQL(ddb: DynamoDbClient, tableNameVal: String, key: String)  
{  
  
    val attDef = AttributeDefinition {  
        attributeName = key  
        attributeType = ScalarAttributeType.N  
    }  
  
    val attDef1 = AttributeDefinition {  
        attributeName = "title"  
        attributeType = ScalarAttributeType.S  
    }  
  
    val keySchemaVal = KeySchemaElement {  
        attributeName = key  
        keyType = KeyType.Hash  
    }  
  
    val keySchemaVal1 = KeySchemaElement {  
        attributeName = "title"  
        keyType = KeyType.Range  
    }  
  
    val provisionedVal = ProvisionedThroughput {  
        readCapacityUnits = 10  
        writeCapacityUnits = 10  
    }  
  
    val request = CreateTableRequest {
```

```
        attributeDefinitions = listOf(attDef, attDef1)
        keySchema = listOf(keySchemaVal, keySchemaVal1)
        provisionedThroughput = provisionedVal
        tableName = tableNameVal
    }

    val response = ddb.createTable(request)
    ddb.waitUntilTableExists { // suspend call
        tableName = tableNameVal
    }
    println("The table was successfully created
    ${response.tableDescription?.tableArn}")
}

suspend fun loadDataPartiQL(ddb: DynamoDbClient, fileName: String) {
    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?, 'info' : ?}"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode
    var t = 0

    while (iter.hasNext()) {

        if (t == 200)
            break

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
        parameters.add(AttributeValue.N(year.toString()))
        parameters.add(AttributeValue.S(title))
        parameters.add(AttributeValue.S(info))

        executeStatementPartiQL(ddb, sqlStatement, parameters)
        println("Added Movie $title")
        parameters.clear()
        t++
    }
}

suspend fun getMoviePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?"
    val parameters: MutableList<AttributeValue> = ArrayList<AttributeValue>()
    parameters.add(AttributeValue.N("2012"))
    parameters.add(AttributeValue.S("The Perks of Being a Wallflower"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun putRecordPartiQL(ddb: DynamoDbClient) {

    val sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?, 'title' : ?, 'info' : ?}"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2020"))
    parameters.add(AttributeValue.S("My Movie"))
    parameters.add(AttributeValue.S("No Info"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Added new movie.")
}
```

```
suspend fun updateTableItemPartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian C. Cooper \",\"Ernest B. Schoedsack\"]' where year=? and title=?"
    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    parameters.add(AttributeValue.S("The East"))
    executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("Item was updated!")
}

// Query the table where the year is 2013.
suspend fun queryTablePartiQL(ddb: DynamoDbClient) {
    val sqlStatement = "SELECT * FROM MoviesPartiQ where year = ?"

    val parameters: MutableList<AttributeValue> = java.util.ArrayList()
    parameters.add(AttributeValue.N("2013"))
    val response = executeStatementPartiQL(ddb, sqlStatement, parameters)
    println("ExecuteStatement successful: $response")
}

suspend fun deleteTablePartiQL(tableNameVal: String) {

    val request = DeleteTableRequest {
        tableName = tableNameVal
    }

    DynamoDbClient { region = "us-east-1" }.use { ddb ->
        ddb.deleteTable(request)
        println("$tableNameVal was deleted")
    }
}

suspend fun executeStatementPartiQL(
    ddb: DynamoDbClient,
    statementVal: String,
    parametersVal: List<AttributeValue>
): ExecuteStatementResponse {

    val request = ExecuteStatementRequest {
        statement = statementVal
        parameters = parametersVal
    }

    return ddb.executeStatement(request)
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Kotlin API reference*.

## Amazon EC2 Auto Scaling examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon EC2 Auto Scaling.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3549\)](#)
- [Scenarios \(p. 3554\)](#)

## Actions

### Create a group

The following code example shows how to create an Auto Scaling group.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createAutoScalingGroup(groupName: String, launchTemplateNameVal: String,  
serviceLinkedRoleARNVal: String, vpcZoneIdVal: String) {  
    val templateSpecification = LaunchTemplateSpecification {  
        launchTemplateName = launchTemplateNameVal  
    }  
  
    val request = CreateAutoScalingGroupRequest {  
        autoScalingGroupName = groupName  
        availabilityZones = listOf("us-east-1a")  
        launchTemplate = templateSpecification  
        maxSize = 1  
        minSize = 1  
        vpcZoneIdentifier = vpcZoneIdVal  
        serviceLinkedRoleArn = serviceLinkedRoleARNVal  
    }  
  
    // This object is required for the waiter call.  
    val groupsRequestWaiter = DescribeAutoScalingGroupsRequest {  
        autoScalingGroupNames = listOf(groupName)  
    }  
  
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->  
        autoScalingClient.createAutoScalingGroup(request)  
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)  
        println("$groupName was created!")  
    }  
}
```

- For API details, see [CreateAutoScalingGroup](#) in *AWS SDK for Kotlin API reference*.

### Delete a group

The following code example shows how to delete an Auto Scaling group.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteSpecificAutoScalingGroup(groupName: String) {  
    val deleteAutoScalingGroupRequest = DeleteAutoScalingGroupRequest {
```

```
        autoScalingGroupName = groupName
        forceDelete = true
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
    }
}
```

- For API details, see [DeleteAutoScalingGroup](#) in *AWS SDK for Kotlin API reference*.

## Disable metrics collection for a group

The following code example shows how to disable CloudWatch metrics collection for an Auto Scaling group.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest = DisableMetricsCollectionRequest {
        autoScalingGroupName = groupName
        metrics = listOf("GroupMaxSize")
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}
```

- For API details, see [DisableMetricsCollection](#) in *AWS SDK for Kotlin API reference*.

## Enable metrics collection for a group

The following code example shows how to enable CloudWatch metrics collection for an Auto Scaling group.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest = EnableMetricsCollectionRequest {
        autoScalingGroupName = groupName
    }
}
```

```
        metrics = listOf("GroupMaxSize")
        granularity = "1Minute"
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

- For API details, see [EnableMetricsCollection](#) in *AWS SDK for Kotlin API reference*.

## Get information about groups

The following code example shows how to get information about Auto Scaling groups.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest = DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}
```

- For API details, see [DescribeAutoScalingGroups](#) in *AWS SDK for Kotlin API reference*.

## Get information about instances

The following code example shows how to get information about Auto Scaling instances.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest {
        instanceIds = listOf(id)
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}
```

- For API details, see [DescribeAutoScalingInstances](#) in *AWS SDK for Kotlin API reference*.

## Get information about scaling activities

The following code example shows how to get information about Auto Scaling activities.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques = DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
        maxRecords = 10
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
${group.healthCheckType}")
        }
    }
}
```

- For API details, see [DescribeScalingActivities](#) in *AWS SDK for Kotlin API reference*.

## Set the desired capacity of a group

The following code example shows how to set the desired capacity of an Auto Scaling group.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest = SetDesiredCapacityRequest {
        autoScalingGroupName = groupName
        desiredCapacity = 2
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}
```

- For API details, see [SetDesiredCapacity](#) in *AWS SDK for Kotlin API reference*.

## Terminate an instance in a group

The following code example shows how to terminate an instance in an Auto Scaling group.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request = TerminateInstanceInAutoScalingGroupRequest {
        instanceId = instanceIdVal
        shouldDecrementDesiredCapacity = false
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- For API details, see [TerminateInstanceInAutoScalingGroup](#) in *AWS SDK for Kotlin API reference*.

## Update a group

The following code example shows how to update the configuration for an Auto Scaling group.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun updateAutoScalingGroup(groupName: String, launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String) {
```

```
val templateSpecification = LaunchTemplateSpecification {  
    launchTemplateName = launchTemplateNameVal  
}  
  
val groupRequest = UpdateAutoScalingGroupRequest {  
    maxSize = 3  
    serviceLinkedRoleArn = serviceLinkedRoleARNVal  
    autoScalingGroupName = groupName  
    launchTemplate = templateSpecification  
}  
  
val groupsRequestWaiter = DescribeAutoScalingGroupsRequest {  
    autoScalingGroupNames = listOf(groupName)  
}  
  
AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->  
    autoScalingClient.updateAutoScalingGroup(groupRequest)  
    autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)  
    println("You successfully updated the Auto Scaling group $groupName")  
}  
}
```

- For API details, see [UpdateAutoScalingGroup in AWS SDK for Kotlin API reference](#).

## Scenarios

### Manage groups and instances

The following code example shows how to:

- Create an Amazon EC2 Auto Scaling group and configure it with a launch template and Availability Zones.
- Get information about the group and running instances.
- Enable Amazon CloudWatch metrics collection on the group.
- Update the desired capacity of the group and wait for an instance to start.
- Terminate an instance in the group.
- List scaling activities that occur in response to user requests and capacity changes.
- Get statistics for CloudWatch metrics that are collected during the example.
- Stop collecting metrics, terminate all instances, and delete the group.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun main(args: Array<String>) {  
    val usage = """  
Usage:  
    <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>  
  
Where:  
    groupName - The name of the Auto Scaling group.  
    launchTemplateName - The name of the launch template.  
    serviceLinkedRoleARN - The ARN of the service-linked role.  
    vpcZoneId - The ID of the VPC zone.  
    """  
    if (args.size != 4) {  
        System.out.println(usage)  
        return  
    }  
    val groupName = args[0]  
    val launchTemplateName = args[1]  
    val serviceLinkedRoleARN = args[2]  
    val vpcZoneId = args[3]  
    // Your logic here  
}  
}
```

```
    serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-linked
    role that the Auto Scaling group uses.
    vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances in
the Auto Scaling group can be created.
    """
    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val groupName = args[0]
    val launchTemplateName = args[1]
    val serviceLinkedRoleARN = args[2]
    val vpcZoneId = args[3]

    println("**** Create an Auto Scaling group named $groupName")
    createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,
    vpcZoneId)

    println("Wait 1 min for the resources, including the instance. Otherwise, an empty
instance Id is returned")
    delay(60000)

    val instanceId = getSpecificAutoScaling(groupName)
    if (instanceId.compareTo("") == 0) {
        println("Error - no instance Id value")
        exitProcess(1)
    } else {
        println("The instance Id value is $instanceId")
    }

    println("**** Describe Auto Scaling with the Id value $instanceId")
    describeAutoScalingInstance(instanceId)

    println("**** Enable metrics collection $instanceId")
    enableMetricsCollection(groupName)

    println("**** Update an Auto Scaling group to maximum size of 3")
    updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

    println("**** Describe all Auto Scaling groups to show the current state of the
groups")
    describeAutoScalingGroups(groupName)

    println("**** Describe account details")
    describeAccountLimits()

    println("Wait 1 min for the resources, including the instance. Otherwise, an empty
instance Id is returned")
    delay(60000)

    println("**** Set desired capacity to 2")
    setDesiredCapacity(groupName)

    println("**** Get the two instance Id values and state")
    getAutoScalingGroups(groupName)

    println("**** List the scaling activities that have occurred for the group")
    describeScalingActivities(groupName)

    println("**** Terminate an instance in the Auto Scaling group")
    terminateInstanceInAutoScalingGroup(instanceId)

    println("**** Stop the metrics collection")
    disableMetricsCollection(groupName)
```

```
    println("**** Delete the Auto Scaling group")
    deleteSpecificAutoScalingGroup(groupName)
}

suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques = DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
        maxRecords = 10
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
${group.healthCheckType}")
        }
    }
}

suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest = DisableMetricsCollectionRequest {
        autoScalingGroupName = groupName
        metrics = listOf("GroupMaxSize")
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest = DescribeScalingActivitiesRequest {
        autoScalingGroupName = groupName
        maxRecords = 10
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
        response.activities?.forEach { activity ->
            println("The activity Id is ${activity.activityId}")
            println("The activity details are ${activity.details}")
        }
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest = DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
        }
    }
}
```

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest = SetDesiredCapacityRequest {
        autoScalingGroupName = groupName
        desiredCapacity = 2
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}

suspend fun updateAutoScalingGroup(groupName: String, launchTemplateNameVal: String,
serviceLinkedRoleARNVal: String) {
    val templateSpecification = LaunchTemplateSpecification {
        launchTemplateName = launchTemplateNameVal
    }

    val groupRequest = UpdateAutoScalingGroupRequest {
        maxSize = 3
        serviceLinkedRoleArn = serviceLinkedRoleARNVal
        autoScalingGroupName = groupName
        launchTemplate = templateSpecification
    }

    val groupsRequestWaiter = DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group $groupName")
    }
}

suspend fun createAutoScalingGroup(groupName: String, launchTemplateNameVal: String,
serviceLinkedRoleARNVal: String, vpcZoneIdVal: String) {
    val templateSpecification = LaunchTemplateSpecification {
        launchTemplateName = launchTemplateNameVal
    }

    val request = CreateAutoScalingGroupRequest {
        autoScalingGroupName = groupName
        availabilityZones = listOf("us-east-1a")
        launchTemplate = templateSpecification
        maxSize = 1
        minSize = 1
        vpcZoneIdentifier = vpcZoneIdVal
        serviceLinkedRoleArn = serviceLinkedRoleARNVal
    }

    // This object is required for the waiter call.
    val groupsRequestWaiter = DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}

suspend fun describeAutoScalingInstance(id: String) {
```

```
    val describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest {
        instanceIds = listOf(id)
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest = EnableMetricsCollectionRequest {
        autoScalingGroupName = groupName
        metrics = listOf("GroupMaxSize")
        granularity = "1Minute"
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}

suspend fun getSpecificAutoScaling(groupName: String): String {
    var instanceId = ""
    val scalingGroupsRequest = DescribeAutoScalingGroupsRequest {
        autoScalingGroupNames = listOf(groupName)
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")

            group.instances?.forEach { instance ->
                instanceId = instance.instanceId.toString()
            }
        }
    }
    return instanceId
}

suspend fun describeAccountLimits() {
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
            autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
        println("The max number of Auto Scaling groups is
${response.maxNumberOfAutoScalingGroups}")
        println("The current number of Auto Scaling groups is
${response.numberOfAutoScalingGroups}")
    }
}

suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request = TerminateInstanceInAutoScalingGroupRequest {
        instanceId = instanceIdVal
        shouldDecrementDesiredCapacity = false
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
    }
}
```

```
        println("You have terminated instance $instanceIdVal")
    }

suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest = DeleteAutoScalingGroupRequest {
        autoScalingGroupName = groupName
        forceDelete = true
    }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.

- [CreateAutoScalingGroup](#)
- [DeleteAutoScalingGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAutoScalingInstances](#)
- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## EventBridge examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon EventBridge.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3559\)](#)

## Actions

### Create a scheduled rule

The following code example shows how to create an Amazon EventBridge scheduled rule.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createScRule(ruleName: String?, cronExpression: String?) {
    val ruleRequest = PutRuleRequest {
        name = ruleName
        eventBusName = "default"
        scheduleExpression = cronExpression
        state = RuleState.Enabled
        description = "A test rule that runs on a schedule created by the Kotlin API"
    }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        val ruleResponse = eventBrClient.putRule(ruleRequest)
        println("The ARN of the new rule is ${ruleResponse.ruleArn}")
    }
}
```

- For API details, see [PutRule](#) in *AWS SDK for Kotlin API reference*.

## Delete a scheduled rule

The following code example shows how to delete an Amazon EventBridge scheduled rule.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteEBRule(ruleName: String) {

    val request = DisableRuleRequest {
        name = ruleName
        eventBusName = "default"
    }

    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->
        eventBrClient.disableRule(request)
        val ruleRequest = DeleteRuleRequest {
            name = ruleName
            eventBusName = "default"
        }

        eventBrClient.deleteRule(ruleRequest)
        println("Rule $ruleName was successfully deleted!")
    }
}
```

- For API details, see [DeleteRule](#) in *AWS SDK for Kotlin API reference*.

## Send events

The following code example shows how to send Amazon EventBridge events.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun putEBEvents(resourceArn: String, resourceArn2: String) {  
  
    // Populate a List with the resource ARN values.  
    val resources0b = mutableListOf<String>()  
    resources0b.add(resourceArn)  
    resources0b.add(resourceArn2)  
  
    val reqEntry = PutEventsRequestEntry {  
        resources = resources0b  
        source = "com.mycompany.myapp"  
        detailType = "myDetailType"  
        detail = "{ \"key1\": \"value1\", \"key2\": \"value2\" }"  
    }  
  
    val request = PutEventsRequest {  
        entries = listOf(reqEntry)  
    }  
  
    EventBridgeClient { region = "us-west-2" }.use { eventBrClient ->  
        val response = eventBrClient.putEvents(request)  
        response.entries?.forEach { resultEntry ->  
  
            if (resultEntry.eventId != null) {  
                println("Event Id is ${resultEntry.eventId}")  
            } else {  
                println("Injection failed with Error Code ${resultEntry.errorCode}")  
            }  
        }  
    }  
}
```

- For API details, see [PutEvents](#) in *AWS SDK for Kotlin API reference*.

## AWS Glue examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with AWS Glue.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3561\)](#)
- [Scenarios \(p. 3564\)](#)

## Actions

### Create a crawler

The following code example shows how to create an AWS Glue crawler.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createGlueCrawler(  
    iam: String?,  
    s3Path: String?,  
    cron: String?,  
    dbName: String?,  
    crawlerName: String  
) {  
    val s3Target = S3Target {  
        path = s3Path  
    }  
  
    // Add the S3Target to a list.  
    val targetList = mutableListOf<S3Target>()  
    targetList.add(s3Target)  
  
    val target0b = CrawlerTargets {  
        s3Targets = targetList  
    }  
  
    val request = CreateCrawlerRequest {  
        databaseName = dbName  
        name = crawlerName  
        description = "Created by the AWS Glue Kotlin API"  
        targets = target0b  
        role = iam  
        schedule = cron  
    }  
  
    GlueClient { region = "us-west-2" }.use { glueClient ->  
        glueClient.createCrawler(request)  
        println("$crawlerName was successfully created")  
    }  
}
```

- For API details, see [CreateCrawler](#) in *AWS SDK for Kotlin API reference*.

## Get a crawler

The following code example shows how to get an AWS Glue crawler.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getSpecificCrawler(crawlerName: String?) {  
    val request = GetCrawlerRequest {
```

```
        name = crawlerName
    }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}
```

- For API details, see [GetCrawler](#) in *AWS SDK for Kotlin API reference*.

## Get a database from the Data Catalog

The following code example shows how to get a database from the AWS Glue Data Catalog.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getSpecificDatabase(databaseName: String?) {

    val request = GetDatabaseRequest {
        name = databaseName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- For API details, see [GetDatabase](#) in *AWS SDK for Kotlin API reference*.

## Start a crawler

The following code example shows how to start an AWS Glue crawler.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun startSpecificCrawler(crawlerName: String?) {

    val request = StartCrawlerRequest {
        name = crawlerName
    }
}
```

```
        GlueClient { region = "us-west-2" }.use { glueClient ->
            glueClient.startCrawler(request)
            println("$crawlerName was successfully started.")
        }
    }
```

- For API details, see [StartCrawler](#) in *AWS SDK for Kotlin API reference*.

## Scenarios

### Get started running crawlers and jobs

The following code example shows how to:

- Create and run a crawler that crawls a public Amazon Simple Storage Service (Amazon S3) bucket and generates a metadata database that describes the CSV-formatted data it finds.
- List information about databases and tables in your AWS Glue Data Catalog.
- Create and run a job that extracts CSV data from the source Amazon S3 bucket, transforms it by removing and renaming fields, and loads JSON-formatted output into another Amazon S3 bucket.
- List information about job runs and view some of the transformed data.
- Delete all resources created by the demo.

For more information, see [Tutorial: Getting started with AWS Glue Studio](#).

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun main(args: Array<String>) {

    val usage = """
        Usage:
            <iام> <s3Path> <cron> <dbName> <crawlerName> <jobName> <scriptLocation>
        <locationUri>

        Where:
            iam - The Amazon Resource Name (ARN) of the AWS Identity and Access
            Management (IAM) role that has AWS Glue and Amazon Simple Storage Service (Amazon S3)
            permissions.
            s3Path - The Amazon Simple Storage Service (Amazon S3) target that contains
            data (for example, CSV data).
            cron - A cron expression used to specify the schedule (for example, cron(15
            12 * * ? *)).
            dbName - The database name.
            crawlerName - The name of the crawler.
            jobName - The name you assign to this job definition.
            scriptLocation - Specifies the Amazon S3 path to a script that runs a job.
            locationUri - Specifies the location of the database
        """

    if (args.size != 8) {
        println(usage)
        exitProcess(1)
```

```
    }

    val iam = args[0]
    val s3Path = args[1]
    val cron = args[2]
    val dbName = args[3]
    val crawlerName = args[4]
    val jobName = args[5]
    val scriptLocation = args[6]
    val locationUri = args[7]

    println("About to start the AWS Glue Scenario")
    createDatabase(dbName, locationUri)
    createCrawler(iam, s3Path, cron, dbName, crawlerName)
    getCrawler(crawlerName)
    startCrawler(crawlerName)
    getDatabase(dbName)
    getGlueTables(dbName)
    createJob(jobName, iam, scriptLocation)
    startJob(jobName)
    getJobs()
    getJobRuns(jobName)
    deleteJob(jobName)
    println("*** Wait for 5 MIN so the $crawlerName is ready to be deleted")
    TimeUnit.MINUTES.sleep(5)
    deleteMyDatabase(dbName)
    deleteCrawler(crawlerName)
}

suspend fun createDatabase(dbName: String?, locationUriVal: String?) {

    val input = DatabaseInput {
        description = "Built with the AWS SDK for Kotlin"
        name = dbName
        locationUri = locationUriVal
    }

    val request = CreateDatabaseRequest {
        databaseInput = input
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createDatabase(request)
        println("The database was successfully created")
    }
}

suspend fun createCrawler(iam: String?, s3Path: String?, cron: String?, dbName: String?, crawlerName: String) {

    val s3Target = S3Target {
        path = s3Path
    }

    val targetList = ArrayList<S3Target>()
    targetList.add(s3Target)

    val target0b = CrawlerTargets {
        s3Targets = targetList
    }

    val crawlerRequest = CreateCrawlerRequest {
        databaseName = dbName
        name = crawlerName
        description = "Created by the AWS Glue Java API"
        targets = target0b
    }
}
```

```
        role = iam
        schedule = cron
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createCrawler(crawlerRequest)
        println("$crawlerName was successfully created")
    }
}

suspend fun getCrawler(crawlerName: String?) {

    val request = GetCrawlerRequest {
        name = crawlerName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}

suspend fun startCrawler(crawlerName: String) {

    val crawlerRequest = StartCrawlerRequest {
        name = crawlerName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.startCrawler(crawlerRequest)
        println("$crawlerName was successfully started.")
    }
}

suspend fun getDatabase(databaseName: String?) {

    val request = GetDatabaseRequest {
        name = databaseName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}

suspend fun getGlueTables(dbName: String?) {

    val tableRequest = GetTablesRequest {
        databaseName = dbName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getTables(tableRequest)
        response.tableList?.forEach { tableName ->
            println("Table name is ${tableName.name}")
        }
    }
}

suspend fun startJob(jobNameVal: String?) {

    val runRequest = StartJobRunRequest {
        workerType = WorkerType.G1X
}
```

```
        numberOfWorkers = 10
        jobName = jobNameVal
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.startJobRun(runRequest)
        println("The job run Id is ${response.jobRunId}")
    }
}

suspend fun createJob(jobName: String, iam: String?, scriptLocationVal: String?) {

    val command0b = JobCommand {
        pythonVersion = "3"
        name = "MyJob1"
        scriptLocation = scriptLocationVal
    }

    val jobRequest = CreateJobRequest {
        description = "A Job created by using the AWS SDK for Java V2"
        glueVersion = "2.0"
        workerType = WorkerType.G1X
        numberOfWorkers = 10
        name = jobName
        role = iam
        command = command0b
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {

    val request = GetJobsRequest {
        maxResults = 10
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobs(request)
        response.jobs?.forEach { job ->
            println("Job name is ${job.name}")
        }
    }
}

suspend fun getJobRuns(jobNameVal: String?) {

    val request = GetJobRunsRequest {
        jobName = jobNameVal
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobRuns(request)
        response.jobRuns?.forEach { job ->
            println("Job name is ${job.jobName}")
        }
    }
}

suspend fun deleteJob(jobNameVal: String) {

    val jobRequest = DeleteJobRequest {
        jobName = jobNameVal
    }
```

```
        }

        GlueClient { region = "us-east-1" }.use { glueClient ->
            glueClient.deleteJob(jobRequest)
            println("$jobNameVal was successfully deleted")
        }
    }

suspend fun deleteMyDatabase(databaseName: String) {

    val request = DeleteDatabaseRequest {
        name = databaseName
    }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteDatabase(request)
        println("$databaseName was successfully deleted")
    }
}

suspend fun deleteCrawler(crawlerName: String) {

    val request = DeleteCrawlerRequest {
        name = crawlerName
    }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.

- [CreateCrawler](#)
- [CreateJob](#)
- [DeleteCrawler](#)
- [DeleteDatabase](#)
- [DeleteJob](#)
- [DeleteTable](#)
- [GetCrawler](#)
- [GetDatabase](#)
- [GetDatabases](#)
- [GetJob](#)
- [GetJobRun](#)
- [GetJobRuns](#)
- [GetTables](#)
- [ListJobs](#)
- [StartCrawler](#)
- [StartJobRun](#)

## IAM examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with AWS Identity and Access Management.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3569\)](#)
- [Scenarios \(p. 3577\)](#)

## Actions

### Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun attachIAMRolePolicy(roleNameVal: String, policyArnVal: String) {  
  
    val request = ListAttachedRolePoliciesRequest {  
        roleName = roleNameVal  
    }  
  
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        val response = iamClient.listAttachedRolePolicies(request)  
        val attachedPolicies = response.attachedPolicies  
  
        // Ensure that the policy is not attached to this role.  
        val checkStatus: Int  
        if (attachedPolicies != null) {  
            checkStatus = checkList(attachedPolicies, policyArnVal)  
            if (checkStatus == -1)  
                return  
        }  
  
        val policyRequest = AttachRolePolicyRequest {  
            roleName = roleNameVal  
            policyArn = policyArnVal  
        }  
        iamClient.attachRolePolicy(policyRequest)  
        println("Successfully attached policy $policyArnVal to role $roleNameVal")  
    }  
}  
  
fun checkList(attachedPolicies: List<AttachedPolicy>, policyArnVal: String): Int {  
  
    for (policy in attachedPolicies) {  
        val polArn = policy.policyArn.toString()  
  
        if (polArn.compareTo(policyArnVal) == 0) {  
            println("The policy is already attached to this role.")  
            return -1  
        }  
    }  
    return 0  
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Kotlin API reference*.

## Create a policy

The following code example shows how to create an IAM policy.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createIAMPolicy(policyNameVal: String?): String {  
  
    val policyDocumentVal = "{" +  
        "  \"Version\": \"2012-10-17\"," +  
        "  \"Statement\": [" +  
        "    {" +  
        "      \"Effect\": \"Allow\"," +  
        "      \"Action\": [" +  
        "        \"dynamodb>DeleteItem\"," +  
        "        \"dynamodb>GetItem\"," +  
        "        \"dynamodb>PutItem\"," +  
        "        \"dynamodb>Scan\"," +  
        "        \"dynamodb>UpdateItem\"" +  
        "      ]," +  
        "      \"Resource\": \"*\" +  
        "    }" +  
        "  ]" +  
    "}"  
  
    val request = CreatePolicyRequest {  
        policyName = policyNameVal  
        policyDocument = policyDocumentVal  
    }  
  
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        val response = iamClient.createPolicy(request)  
        return response.policy?.arn.toString()  
    }  
}
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Kotlin API reference*.

## Create a user

The following code example shows how to create an IAM user.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createIAMUser(usernameVal: String?): String? {  
  
    val request = CreateUserRequest {  
        userName = usernameVal  
    }  
  
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        val response = iamClient.createUser(request)  
        return response.user?.userName  
    }  
}
```

- For API details, see [CreateUser](#) in *AWS SDK for Kotlin API reference*.

## Create an access key

The following code example shows how to create an IAM access key.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createIAMAccessKey(user: String?): String {  
  
    val request = CreateAccessKeyRequest {  
        userName = user  
    }  
  
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        val response = iamClient.createAccessKey(request)  
        return response.accessKey?.accessKeyId.toString()  
    }  
}
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for Kotlin API reference*.

## Create an alias for an account

The following code example shows how to create an alias for an IAM account.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createIAMAccountAlias(alias: String) {
```

```
    val request = CreateAccountAliasRequest {
        accountAlias = alias
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- For API details, see [CreateAccountAlias](#) in *AWS SDK for Kotlin API reference*.

## Delete a policy

The following code example shows how to delete an IAM policy.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {

    val request = DeletePolicyRequest {
        policyArn = policyARNVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- For API details, see [DeletePolicy](#) in *AWS SDK for Kotlin API reference*.

## Delete a user

The following code example shows how to delete an IAM user.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteIAMUser(userNameVal: String) {

    val request = DeleteUserRequest {
        userName = userNameVal
    }
}
```

```
// To delete a user, ensure that the user's access keys are deleted first.  
IamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
    iamClient.deleteUser(request)  
    println("Successfully deleted user $userNameVal")  
}  
}
```

- For API details, see [DeleteUser](#) in *AWS SDK for Kotlin API reference*.

## Delete an access key

The following code example shows how to delete an IAM access key.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteKey(userNameVal: String, accessKey: String) {  
  
    val request = DeleteAccessKeyRequest {  
        accessKeyId = accessKey  
        userName = userNameVal  
    }  
  
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        iamClient.deleteAccessKey(request)  
        println("Successfully deleted access key $accessKey from $userNameVal")  
    }  
}
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for Kotlin API reference*.

## Delete an account alias

The following code example shows how to delete an IAM account alias.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteIAMAccountAlias(alias: String) {  
  
    val request = DeleteAccountAliasRequest {  
        accountAlias = alias  
    }  
}
```

```
IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
    iamClient.deleteAccountAlias(request)
    println("Successfully deleted account alias $alias")
}
```

- For API details, see [DeleteAccountAlias](#) in *AWS SDK for Kotlin API reference*.

## Detach a policy from a role

The following code example shows how to detach an IAM policy from a role.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun detachPolicy(roleNameVal: String, policyArnVal: String) {

    val request = DetachRolePolicyRequest {
        roleName = roleNameVal
        policyArn = policyArnVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")
    }
}
```

- For API details, see [DetachRolePolicy](#) in *AWS SDK for Kotlin API reference*.

## Get a policy

The following code example shows how to get an IAM policy.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getIAMPolicy(policyArnVal: String?) {

    val request = GetPolicyRequest {
        policyArn = policyArnVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
    }
}
```

```
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- For API details, see [GetPolicy](#) in *AWS SDK for Kotlin API reference*.

## List a user's access keys

The following code example shows how to list a user's IAM access keys.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listKeys(userNameVal: String?) {

    val request = ListAccessKeysRequest {
        userName = userNameVal
    }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- For API details, see [ListAccessKeys](#) in *AWS SDK for Kotlin API reference*.

## List account aliases

The following code example shows how to list IAM account aliases.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAliases() {

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- For API details, see [ListAccountAliases](#) in *AWS SDK for Kotlin API reference*.

## List users

The following code example shows how to list IAM users.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAllUsers() {  
  
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        val response = iamClient.listUsers(ListUsersRequest { })  
        response.users?.forEach { user ->  
            println("Retrieved user ${user.userName}")  
            val permissionsBoundary = user.permissionsBoundary  
            if (permissionsBoundary != null)  
                println("Permissions boundary details  
${permissionsBoundary.permissionsBoundaryType}")  
        }  
    }  
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Kotlin API reference*.

## Update a user

The following code example shows how to update an IAM user.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun updateIAMUser(curName: String?, newName: String?) {  
  
    val request = UpdateUserRequest {  
        userName = curName  
        newUserName = newName  
    }  
  
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        iamClient.updateUser(request)  
        println("Successfully updated user to $newName")  
    }  
}
```

- For API details, see [UpdateUser](#) in *AWS SDK for Kotlin API reference*.

## Scenarios

### Create a user and assume a role

The following code example shows how to:

- Create a user who has no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list Amazon S3 buckets using temporary credentials.
- Delete the policy, role, and user.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap IAM user actions.

```
suspend fun main(args: Array<String>) {  
  
    val usage = """  
Usage:  
    <username> <policyName> <roleName> <roleSessionName> <fileLocation>  
<bucketName>  
  
Where:  
    username - The name of the IAM user to create.  
    policyName - The name of the policy to create.  
    roleName - The name of the role to create.  
    roleSessionName - The name of the session required for the assumeRole  
operation.  
    fileLocation - The file location to the JSON required to create the role (see  
Readme).  
    bucketName - The name of the Amazon S3 bucket from which objects are read.  
    """  
  
    if (args.size != 6) {  
        println(usage)  
        exitProcess(1)  
    }  
  
    val userName = args[0]  
    val policyName = args[1]  
    val roleName = args[2]  
    val roleSessionName = args[3]  
    val fileLocation = args[4]  
    val bucketName = args[5]  
  
    createUser(userName)  
    println("$userName was successfully created.")  
  
    val polArn = createPolicy(policyName)  
    println("The policy $polArn was successfully created.")
```

```
    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("/** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)

    println("/** Getting ready to delete the AWS resources.")
    deleteRole(roleName, polArn)
    deleteUser(userName)
    println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {

    val request = CreateUserRequest {
        userName = usernameVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}

suspend fun createPolicy(policyNameVal: String?): String {

    val policyDocumentValue: String = "{" +
        " \"Version\": \"2012-10-17\", " +
        " \"Statement\": [ " +
        "   { " +
        "     \"Effect\": \"Allow\", " +
        "     \"Action\": [ " +
        "       \"s3:*\" " +
        "     ], " +
        "     \"Resource\": \"*\" " +
        "   } " +
        " ] " +
        "}"

    val request = CreatePolicyRequest {
        policyName = policyNameVal
        policyDocument = policyDocumentValue
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(rolenameVal: String?, fileLocation: String?): String? {

    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request = CreateRoleRequest {
        roleName = rolenameVal
        assumeRolePolicyDocument = jsonObject.toJSONString()
        description = "Created using the AWS SDK for Kotlin"
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}
```

```
        }

    suspend fun attachRolePolicy(roleNameVal: String, policyArnVal: String) {

        val request = ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

        IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
            val response = iamClient.listAttachedRolePolicies(request)
            val attachedPolicies = response.attachedPolicies

            // Ensure that the policy is not attached to this role.
            val checkStatus: Int
            if (attachedPolicies != null) {
                checkStatus = checkMyList(attachedPolicies, policyArnVal)
                if (checkStatus == -1)
                    return
            }

            val policyRequest = AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
            iamClient.attachRolePolicy(policyRequest)
            println("Successfully attached policy $policyArnVal to role $roleNameVal")
        }
    }

    fun checkMyList(attachedPolicies: List<AttachedPolicy>, policyArnVal: String): Int {

        for (policy in attachedPolicies) {
            val polArn = policy.policyArn.toString()

            if (polArn.compareTo(policyArnVal) == 0) {
                println("The policy is already attached to this role.")
                return -1
            }
        }
        return 0
    }

    suspend fun assumeGivenRole(roleArnVal: String?, roleSessionNameVal: String?,
                                bucketName: String) {

        val stsClient = StsClient {
            region = "us-east-1"
        }

        val roleRequest = AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

        val roleResponse = stsClient.assumeRole(roleRequest)
        val myCreds = roleResponse.credentials
        val key = myCreds?.accessKeyId
        val secKey = myCreds?.secretAccessKey
        val secToken = myCreds?.sessionToken

        val staticCredentials = StaticCredentialsProvider {
            accessKeyId = key
            secretAccessKey = secKey
            sessionToken = secToken
        }
    }
}
```

```
// List all objects in an Amazon S3 bucket using the temp creds.
val s3 = S3Client {
    credentialsProvider = staticCredentials
    region = "us-east-1"
}

println("Created a S3Client using temp credentials.")
println("Listing objects in $bucketName")

val listObjects = ListObjectsRequest {
    bucket = bucketName
}

val response = s3.listObjects(listObjects)
response.contents?.forEach { myObject ->
    println("The name of the key is ${myObject.key}")
    println("The owner is ${myObject.owner}")
}
}

suspend fun deleteRole(roleNameVal: String, polArn: String) {

    val iam = IamClient { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest = DetachRolePolicyRequest {
        policyArn = polArn
        roleName = roleNameVal
    }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request = DeletePolicyRequest {
        policyArn = polArn
    }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest = DeleteRoleRequest {
        roleName = roleNameVal
    }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient { region = "AWS_GLOBAL" }
    val request = DeleteUserRequest {
        userName = userNameVal
    }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## AWS KMS examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with AWS Key Management Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3581\)](#)

## Actions

### Create a grant for a key

The following code example shows how to create a grant for a KMS key.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createNewGrant(keyIdVal: String?, granteePrincipalVal: String?, operation: String): String? {  
  
    val operation0b = GrantOperation.fromValue(operation)  
    val grantOperationList = ArrayList<GrantOperation>()  
    grantOperationList.add(operation0b)  
  
    val request = CreateGrantRequest {  
        keyId = keyIdVal  
        granteePrincipal = granteePrincipalVal  
        operations = grantOperationList  
    }  
  
    return client.createGrant(request).grantId  
}
```

```
        }

        KmsClient { region = "us-west-2" }.use { kmsClient ->
            val response = kmsClient.createGrant(request)
            return response.grantId
        }
    }
```

- For API details, see [CreateGrant](#) in *AWS SDK for Kotlin API reference*.

## Create a key

The following code example shows how to create an AWS KMS key.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createKey(keyDesc: String?): String? {

    val request = CreateKeyRequest {
        description = keyDesc
        customerMasterKeySpec = CustomerMasterKeySpec.SymmetricDefault
        keyUsage = KeyUsageType.fromValue("ENCRYPT_DECRYPT")
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val result = kmsClient.createKey(request)
        println("Created a customer key with id " + result.keyMetadata?.arn)
        return result.keyMetadata?.keyId
    }
}
```

- For API details, see [CreateKey](#) in *AWS SDK for Kotlin API reference*.

## Create an alias for a key

The following code example shows how to create an alias for a KMS key.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createCustomAlias(targetKeyIdVal: String?, aliasNameVal: String?) {

    val request = CreateAliasRequest {
        aliasName = aliasNameVal
        targetKeyId = targetKeyIdVal
    }
}
```

```
KmsClient { region = "us-west-2" }.use { kmsClient ->
    kmsClient.createAlias(request)
    println("$aliasNameVal was successfully created")
}
}
```

- For API details, see [CreateAlias](#) in *AWS SDK for Kotlin API reference*.

## Decrypt ciphertext

The following code example shows how to decrypt ciphertext that was encrypted by a KMS key.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun encryptData(keyIdValue: String): ByteArray? {
    val text = "This is the text to encrypt by using the AWS KMS Service"
    val myBytes: ByteArray = text.toByteArray()

    val encryptRequest = EncryptRequest {
        keyId = keyIdValue
        plaintext = myBytes
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.encrypt(encryptRequest)
        val algorithm: String = response.encryptionAlgorithm.toString()
        println("The encryption algorithm is $algorithm")

        // Return the encrypted data.
        return response.ciphertextBlob
    }
}

suspend fun decryptData(encryptedDataVal: ByteArray?, keyIdVal: String?, path: String)?
{
    val decryptRequest = DecryptRequest {
        ciphertextBlob = encryptedDataVal
        keyId = keyIdVal
    }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Write the decrypted data to a file.
        if (myVal != null) {
            File(path).writeBytes(myVal)
        }
    }
}
```

- For API details, see [Decrypt](#) in *AWS SDK for Kotlin API reference*.

## Describe a key

The following code example shows how to describe a KMS key.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeSpecifcKey(keyIdVal: String?) {  
  
    val request = DescribeKeyRequest {  
        keyId = keyIdVal  
    }  
  
    KmsClient { region = "us-west-2" }.use { kmsClient ->  
        val response = kmsClient.describeKey(request)  
        println("The key description is ${response.keyMetadata?.description}")  
        println("The key ARN is ${response.keyMetadata?.arn}")  
    }  
}
```

- For API details, see [DescribeKey](#) in *AWS SDK for Kotlin API reference*.

## Disable a key

The following code example shows how to disable a KMS key.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun disableKey(keyIdVal: String?) {  
  
    val request = DisableKeyRequest {  
        keyId = keyIdVal  
    }  
  
    KmsClient { region = "us-west-2" }.use { kmsClient ->  
        kmsClient.disableKey(request)  
        println("$keyIdVal was successfully disabled")  
    }  
}
```

- For API details, see [DisableKey](#) in *AWS SDK for Kotlin API reference*.

## Enable a key

The following code example shows how to enable a KMS key.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun enableKey(keyIdVal: String?) {  
  
    val request = EnableKeyRequest {  
        keyId = keyIdVal  
    }  
  
    KmsClient { region = "us-west-2" }.use { kmsClient ->  
        kmsClient.enableKey(request)  
        println("$keyIdVal was successfully enabled.")  
    }  
}
```

- For API details, see [EnableKey](#) in *AWS SDK for Kotlin API reference*.

## Encrypt text using a key

The following code example shows how to encrypt text using a KMS key.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun encryptData(keyIdValue: String): ByteArray? {  
  
    val text = "This is the text to encrypt by using the AWS KMS Service"  
    val myBytes: ByteArray = text.toByteArray()  
  
    val encryptRequest = EncryptRequest {  
        keyId = keyIdValue  
        plaintext = myBytes  
    }  
  
    KmsClient { region = "us-west-2" }.use { kmsClient ->  
        val response = kmsClient.encrypt(encryptRequest)  
        val algorithm: String = response.encryptionAlgorithm.toString()  
        println("The encryption algorithm is $algorithm")  
  
        // Return the encrypted data.  
        return response.ciphertextBlob  
    }  
}  
  
suspend fun decryptData(encryptedDataVal: ByteArray?, keyIdVal: String?, path: String)  
{  
  
    val decryptRequest = DecryptRequest {  
        ciphertextBlob = encryptedDataVal  
    }
```

```
        keyId = keyIdVal
    }
    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val decryptResponse = kmsClient.decrypt(decryptRequest)
        val myVal = decryptResponse.plaintext

        // Write the decrypted data to a file.
        if (myVal != null) {
            File(path).writeBytes(myVal)
        }
    }
}
```

- For API details, see [Encrypt in AWS SDK for Kotlin API reference](#).

## List aliases for a key

The following code example shows how to list aliases for a KMS key.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAllAliases() {

    val request = ListAliasesRequest {
        limit = 15
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listAliases(request)
        response.aliases?.forEach { alias ->
            println("The alias name is ${alias.aliasName}")
        }
    }
}
```

- For API details, see [ListAliases in AWS SDK for Kotlin API reference](#).

## List grants for a key

The following code example shows how to list grants for a KMS key.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun displayGrantIds(keyIdVal: String?) {
```

```
    val request = ListGrantsRequest {
        keyId = keyIdVal
        limit = 15
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listGrants(request)
        response.grants?.forEach { grant ->
            println("The grant Id is ${grant.grantId}")
        }
    }
}
```

- For API details, see [ListGrants](#) in *AWS SDK for Kotlin API reference*.

## List keys

The following code example shows how to list KMS keys.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAllKeys() {

    val request = ListKeysRequest {
        limit = 15
    }

    KmsClient { region = "us-west-2" }.use { kmsClient ->
        val response = kmsClient.listKeys(request)
        response.keys?.forEach { key ->
            println("The key ARN is ${key.keyArn}")
            println("The key Id is ${key.keyId}")
        }
    }
}
```

- For API details, see [ListKeys](#) in *AWS SDK for Kotlin API reference*.

## Lambda examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with AWS Lambda.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3588\)](#)
- [Scenarios \(p. 3589\)](#)

## Actions

### Create a function

The following code example shows how to create a Lambda function.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createNewFunction(  
    myFunctionName: String,  
    s3BucketName: String,  
    myS3Key: String,  
    myHandler: String,  
    myRole: String  
) : String? {  
  
    val functionCode = FunctionCode {  
        s3Bucket = s3BucketName  
        s3Key = myS3Key  
    }  
  
    val request = CreateFunctionRequest {  
        functionName = myFunctionName  
        code = functionCode  
        description = "Created by the Lambda Kotlin API"  
        handler = myHandler  
        role = myRole  
        runtime = Runtime.Java8  
    }  
  
    LambdaClient { region = "us-west-2" }.use { awsLambda ->  
        val functionResponse = awsLambda.createFunction(request)  
        awsLambda.waitUntilFunctionActive {  
            functionName = myFunctionName  
        }  
        return functionResponse.functionArn  
    }  
}
```

- For API details, see [CreateFunction](#) in *AWS SDK for Kotlin API reference*.

### Delete a function

The following code example shows how to delete a Lambda function.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteLambdaFunction(myFunctionName: String) {  
  
    val request = DeleteFunctionRequest {  
        functionName = myFunctionName  
    }  
  
    LambdaClient { region = "us-west-2" }.use { awsLambda ->  
        awsLambda.deleteFunction(request)  
        println("$myFunctionName was deleted")  
    }  
}
```

- For API details, see [DeleteFunction](#) in *AWS SDK for Kotlin API reference*.

## Invoke a function

The following code example shows how to invoke a Lambda function.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun invokeFunction(functionNameVal: String) {  
  
    val json = """{"inputValue":"1000"}"""  
    val byteArray = json.trimIndent().encodeToByteArray()  
    val request = InvokeRequest {  
        functionName = functionNameVal  
        logType = LogType.Tail  
        payload = byteArray  
    }  
  
    LambdaClient { region = "us-west-2" }.use { awsLambda ->  
        val res = awsLambda.invoke(request)  
        println("${res.payload?.toString(Charsets.UTF_8)}")  
        println("The log result is ${res.logResult}")  
    }  
}
```

- For API details, see [Invoke](#) in *AWS SDK for Kotlin API reference*.

## Scenarios

### Get started with functions

The following code example shows how to:

- Create an AWS Identity and Access Management (IAM) role that grants Lambda permission to write to logs.
- Create a Lambda function and upload handler code.
- Invoke the function with a single parameter and get results.

- Update the function code and configure its Lambda environment with an environment variable.
- Invoke the function with new parameters and get results. Display the execution log that's returned from the invocation.
- List the functions for your account.
- Delete the IAM role and the Lambda function.

For more information, see [Create a Lambda function with the console](#).

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun main(args: Array<String>) {  
  
    val usage = """  
        Usage:  
            <functionName> <role> <handler> <bucketName> <updatedBucketName> <key>  
  
        Where:  
            functionName - The name of the AWS Lambda function.  
            role - The AWS Identity and Access Management (IAM) service role that has  
AWS Lambda permissions.  
            handler - The fully qualified method name (for example,  
example.Handler::handleRequest).  
            bucketName - The Amazon Simple Storage Service (Amazon S3) bucket name that  
contains the ZIP or JAR used for the Lambda function's code.  
            updatedBucketName - The Amazon S3 bucket name that contains the .zip  
or .jar used to update the Lambda function's code.  
            key - The Amazon S3 key name that represents the .zip or .jar file (for  
example, LambdaHello-1.0-SNAPSHOT.jar).  
        """  
  
    if (args.size != 6) {  
        println(usage)  
        exitProcess(1)  
    }  
  
    val functionName = args[0]  
    val role = args[1]  
    val handler = args[2]  
    val bucketName = args[3]  
    val updatedBucketName = args[4]  
    val key = args[5]  
  
    println("Creating a Lambda function named $functionName.")  
    val funArn = createScFunction(functionName, bucketName, key, handler, role)  
    println("The AWS Lambda ARN is $funArn")  
  
    // Get a specific Lambda function.  
    println("Getting the $functionName AWS Lambda function.")  
    getFunction(functionName)  
  
    // List the Lambda functions.  
    println("Listing all AWS Lambda functions.")  
    listFunctionsSc()  
  
    // Invoke the Lambda function.
```

```
    println("/** Invoke the Lambda function.")  
    invokeFunctionSc(functionName)  
  
    // Update the AWS Lambda function code.  
    println("/** Update the Lambda function code.")  
    updateFunctionCode(functionName, updatedBucketName, key)  
  
    // println("/** Invoke the function again after updating the code.")  
    invokeFunctionSc(functionName)  
  
    // Update the AWS Lambda function configuration.  
    println("Update the run time of the function.")  
    UpdateFunctionConfiguration(functionName, handler)  
  
    // Delete the AWS Lambda function.  
    println("Delete the AWS Lambda function.")  
    delFunction(functionName)  
}  
  
suspend fun createScFunction(  
    myFunctionName: String,  
    s3BucketName: String,  
    myS3Key: String,  
    myHandler: String,  
    myRole: String  
): String {  
  
    val functionCode = FunctionCode {  
        s3Bucket = s3BucketName  
        s3Key = myS3Key  
    }  
  
    val request = CreateFunctionRequest {  
        functionName = myFunctionName  
        code = functionCode  
        description = "Created by the Lambda Kotlin API"  
        handler = myHandler  
        role = myRole  
        runtime = Runtime.Java8  
    }  
  
    // Create a Lambda function using a waiter  
    LambdaClient { region = "us-west-2" }.use { awsLambda ->  
        val functionResponse = awsLambda.createFunction(request)  
        awsLambda.waitUntilFunctionActive {  
            functionName = myFunctionName  
        }  
        return functionResponse.functionArn.toString()  
    }  
}  
  
suspend fun getFunction(functionNameVal: String) {  
  
    val functionRequest = GetFunctionRequest {  
        functionName = functionNameVal  
    }  
  
    LambdaClient { region = "us-west-2" }.use { awsLambda ->  
        val response = awsLambda.getFunction(functionRequest)  
        println("The runtime of this Lambda function is  
        ${response.configuration?.runtime}")  
    }  
}  
  
suspend fun listFunctionsSc() {
```

```
    val request = ListFunctionsRequest {
        maxItems = 10
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val response = awsLambda.listFunctions(request)
        response.functions?.forEach { function ->
            println("The function name is ${function.functionName}")
        }
    }
}

suspend fun invokeFunctionSc(functionNameVal: String) {

    val json = """{"inputValue":"1000"}"""
    val byteArray = json.trimIndent().encodeToByteArray()
    val request = InvokeRequest {
        functionName = functionNameVal
        payload = byteArray
        logType = LogType.Tail
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val res = awsLambda.invoke(request)
        println("The function payload is ${res.payload?.toString(Charsets.UTF_8)}")
    }
}

suspend fun updateFunctionCode(functionNameVal: String?, bucketName: String?, key: String?) {

    val functionCodeRequest = UpdateFunctionCodeRequest {
        functionName = functionNameVal
        publish = true
        s3Bucket = bucketName
        s3Key = key
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        val response = awsLambda.updateFunctionCode(functionCodeRequest)
        awsLambda.waitUntilFunctionUpdated {
            functionName = functionNameVal
        }
        println("The last modified value is " + response.lastModified)
    }
}

suspend fun UpdateFunctionConfiguration(functionNameVal: String?, handlerVal: String?) {

    val configurationRequest = UpdateFunctionConfigurationRequest {
        functionName = functionNameVal
        handler = handlerVal
        runtime = Runtime.Java11
    }

    LambdaClient { region = "us-west-2" }.use { awsLambda ->
        awsLambda.updateFunctionConfiguration(configurationRequest)
    }
}

suspend fun delFunction(myFunctionName: String) {

    val request = DeleteFunctionRequest {
        functionName = myFunctionName
    }
```

```
LambdaClient { region = "us-west-2" }.use { awsLambda ->
    awsLambda.deleteFunction(request)
    println("$myFunctionName was deleted")
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## Amazon Pinpoint examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Pinpoint.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3593\)](#)

## Actions

### Create a campaign

The following code example shows how to create a campaign.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createPinCampaign(appId: String, segmentIdVal: String) {

    val schedule0b = Schedule {
        startTime = "IMMEDIATE"
    }

    val defaultMessage0b = Message {
        action = Action.OpenApp
        body = "My message body"
        title = "My message title"
    }
}
```

```
    val messageConfiguration0b = MessageConfiguration {
        defaultMessage = defaultMessage0b
    }

    val writeCampaign = WriteCampaignRequest {
        description = "My description"
        schedule = schedule0b
        name = "MyCampaign"
        segmentId = segmentIdVal
        messageConfiguration = messageConfiguration0b
    }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result: CreateCampaignResponse = pinpoint.createCampaign(
            CreateCampaignRequest {
                applicationId = appId
                writeCampaignRequest = writeCampaign
            }
        )
        println("Campaign ID is ${result.campaignResponse?.id}")
    }
}
```

- For API details, see [CreateCampaign in AWS SDK for Kotlin API reference](#).

## Create a segment

The following code example shows how to create a segment.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createPinpointSegment(applicationIdVal: String?): String? {

    val segmentAttributes = mutableMapOf<String, AttributeDimension>()
    val myList = mutableListOf<String>()
    myList.add("Lakers")

    val atts = AttributeDimension {
        attributeType = AttributeType.Inclusive
        values = myList
    }

    segmentAttributes["Team"] = atts
    val recencyDimension = RecencyDimension {
        duration = Duration.fromValue("DAY_30")
        recencyType = RecencyType.fromValue("ACTIVE")
    }

    val segmentBehaviors = SegmentBehaviors {
        recency = recencyDimension
    }

    val segmentLocation = SegmentLocation {}
    val dimensions0b = SegmentDimensions {
```

```
        attributes = segmentAttributes
        behavior = segmentBehaviors
        demographic = SegmentDemographics {}
        location = segmentLocation
    }

    val writeSegmentRequest0b = WriteSegmentRequest {
        name = "MySegment101"
        dimensions = dimensions0b
    }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val createSegmentResult: CreateSegmentResponse = pinpoint.createSegment(
            CreateSegmentRequest {
                applicationId = applicationIdVal
                writeSegmentRequest = writeSegmentRequest0b
            }
        )
        println("Segment ID is ${createSegmentResult.segmentResponse?.id}")
        return createSegmentResult.segmentResponse?.id
    }
}
```

- For API details, see [CreateSegment in AWS SDK for Kotlin API reference](#).

## Create an application

The following code example shows how to create an application.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createApplication(applicationName: String?): String? {

    val createApplicationRequest0b = CreateApplicationRequest {
        name = applicationName
    }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        val result = pinpoint.createApp(
            CreateAppRequest {
                createApplicationRequest = createApplicationRequest0b
            }
        )
        return result.applicationResponse?.id
    }
}
```

- For API details, see [CreateApp in AWS SDK for Kotlin API reference](#).

## Delete an application

The following code example shows how to delete an application.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deletePinApp(appId: String?) {  
  
    PinpointClient { region = "us-west-2" }.use { pinpoint ->  
        val result = pinpoint.deleteApp(  
            DeleteAppRequest {  
                applicationId = appId  
            }  
        )  
        val appName = result.applicationResponse?.name  
        println("Application $appName has been deleted.")  
    }  
}
```

- For API details, see [DeleteApp](#) in *AWS SDK for Kotlin API reference*.

## Delete an endpoint

The following code example shows how to delete an endpoint.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deletePinEndpoint(appIdVal: String?, endpointIdVal: String?) {  
  
    val deleteEndpointRequest = DeleteEndpointRequest {  
        applicationId = appIdVal  
        endpointId = endpointIdVal  
    }  
  
    PinpointClient { region = "us-west-2" }.use { pinpoint ->  
        val result = pinpoint.deleteEndpoint(deleteEndpointRequest)  
        val id = result.endpointResponse?.id  
        println("The deleted endpoint is $id")  
    }  
}
```

- For API details, see [DeleteEndpoint](#) in *AWS SDK for Kotlin API reference*.

## Get endpoints

The following code example shows how to get endpoints.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun lookupPinpointEndpoint(appId: String?, endpoint: String?) {  
  
    PinpointClient { region = "us-west-2" }.use { pinpoint ->  
        val result = pinpoint.getEndpoint(  
            GetEndpointRequest {  
                applicationId = appId  
                endpointId = endpoint  
            }  
        )  
        val endResponse = result.endpointResponse  
  
        // Uses the Google Gson library to pretty print the endpoint JSON.  
        val gson: com.google.gson.Gson = GsonBuilder()  
            .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)  
            .setPrettyPrinting()  
            .create()  
  
        val endpointJson: String = gson.toJson(endResponse)  
        println(endpointJson)  
    }  
}
```

- For API details, see [GetEndpoint](#) in *AWS SDK for Kotlin API reference*.

## List segments

The following code example shows how to list segments.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listSegs(appId: String?) {  
  
    PinpointClient { region = "us-west-2" }.use { pinpoint ->  
  
        val response = pinpoint.getSegments(  
            GetSegmentsRequest {  
                applicationId = appId  
            }  
        )  
        response.segmentsResponse?.item?.forEach { segment ->  
            println("Segment id is ${segment.id}")  
        }  
    }  
}
```

- For API details, see [GetSegments](#) in *AWS SDK for Kotlin API reference*.

## Send email and text messages

The following code example shows how to send email and text messages with Amazon Pinpoint.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun sendEmail(  
    msgSubject: String?,  
    appId: String?,  
    senderAddress: String?,  
    toAddress: String  
) {  
  
    // The body of the email for recipients whose email clients support HTML content.  
    val htmlBody = (  
        "<h1>Amazon Pinpoint test (AWS SDK for Kotlin)</h1>" +  
        "<p>This email was sent through the <a href='https://aws.amazon.com/"  
        "pinpoint/'>" +  
        "Amazon Pinpoint</a> Email API"  
    )  
  
    // The character encoding to use for the subject line and the message body.  
    val charsetVal = "UTF-8"  
  
    val addressMap = mutableMapOf<String, AddressConfiguration>()  
    val configuration = AddressConfiguration {  
        channelType = ChannelType.Email  
    }  
  
    addressMap[toAddress] = configuration  
    val emailPart = SimpleEmailPart {  
        data = htmlBody  
        charset = charsetVal  
    }  
  
    val subjectPart0b = SimpleEmailPart {  
        data = msgSubject  
        charset = charsetVal  
    }  
  
    val simpleEmail0b = SimpleEmail {  
        htmlPart = emailPart  
        subject = subjectPart0b  
    }  
  
    val emailMessage0b = EmailMessage {  
        body = htmlBody  
        fromAddress = senderAddress  
        simpleEmail = simpleEmail0b  
    }  
  
    val directMessageConfiguration0b = DirectMessageConfiguration {
```

```
        emailMessage = emailMessage0b
    }

    val messageRequest0b = MessageRequest {
        addresses = addressMap
        messageConfiguration = directMessageConfiguration0b
    }

    PinpointClient { region = "us-west-2" }.use { pinpoint ->
        pinpoint.sendMessages(
            SendMessagesRequest {
                applicationId = appId
                messageRequest = messageRequest0b
            }
        )
        println("The email message was successfully sent")
    }
}
```

- For API details, see [SendMessages](#) in *AWS SDK for Kotlin API reference*.

## Amazon RDS examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Relational Database Service.

**Actions** are code excerpts that show you how to call individual service functions.

**Scenarios** are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3599\)](#)
- [Scenarios \(p. 3602\)](#)

## Actions

### Create a DB instance

The following code example shows how to create an Amazon RDS DB instance and wait for it to become available.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createDatabaseInstance(
    dbInstanceIdentifierVal: String?,
    dbNamedbVal: String?,
    masterUsernameVal: String?,
    masterUserPasswordVal: String?
) {

    val instanceRequest = CreateDbInstanceRequest {
```

```
        dbInstanceIdentifier = dbInstanceIdentifierVal
        allocatedStorage = 100
        dbName = dbNameVal
        engine = "mysql"
        dbInstanceClass = "db.m4.large"
        engineVersion = "8.0.15"
        storageType = "standard"
        masterUsername = masterUsernameVal
        masterUserPassword = masterUserPasswordVal
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceState}")
    }
}

// Waits until the database instance is available.
suspend fun waitForInstanceReady(dbInstanceIdentifierVal: String?) {
    val sleepTime: Long = 20
    var instanceReady = false
    var instanceReadyStr = ""
    println("Waiting for instance to become available.")

    val instanceRequest = DescribeDbInstancesRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {

                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceState.toString()
                    if (instanceReadyStr.contains("available"))
                        instanceReady = true
                    else {
                        println("...$instanceReadyStr")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
        println("Database instance is available!")
    }
}
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for Kotlin API reference*.

## Delete a DB instance

The following code example shows how to delete an Amazon RDS DB instance.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteDatabaseInstance(dbInstanceIdentifierVal: String?) {  
  
    val deleteDbInstanceRequest = DeleteDbInstanceRequest {  
        dbInstanceIdentifier = dbInstanceIdentifierVal  
        deleteAutomatedBackups = true  
        skipFinalSnapshot = true  
    }  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)  
        print("The status of the database is ${response.dbInstance?.dbInstanceState}")  
    }  
}
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for Kotlin API reference*.

## Describe DB instances

The following code example shows how to describe Amazon RDS DB instances.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeInstances() {  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val response = rdsClient.describeDbInstances(DescribeDbInstancesRequest {})  
        response.dbInstances?.forEach { instance ->  
            println("Instance Identifier is ${instance.dbInstanceIdentifier}")  
            println("The Engine is ${instance.engine}")  
            println("Connection endpoint is ${instance.endpoint?.address}")  
        }  
    }  
}
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for Kotlin API reference*.

## Modify a DB instance

The following code example shows how to modify an Amazon RDS DB instance.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun updateInstance(dbInstanceIdentifierVal: String?, masterUserPasswordVal: String?) {  
  
    val request = ModifyDbInstanceRequest {  
        dbInstanceIdentifier = dbInstanceIdentifierVal  
        publiclyAccessible = true  
        masterUserPassword = masterUserPasswordVal  
    }  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val instanceResponse = rdsClient.modifyDbInstance(request)  
        println("The ARN of the modified database is  
        ${instanceResponse.dbInstance?.dbInstanceArn}")  
    }  
}
```

- For API details, see [ModifyDBInstance](#) in *AWS SDK for Kotlin API reference*.

## Retrieve attributes

The following code example shows how to retrieve attributes that belong to an Amazon RDS account.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getAccountAttributes() {  
  
    RdsClient { region = "us-west-2" }.use { rdsClient ->  
        val response =  
        rdsClient.describeAccountAttributes(DescribeAccountAttributesRequest {})  
        response.accountQuotas?.forEach { quotas ->  
            val response = response.accountQuotas  
            println("Name is: ${quotas.accountQuotaName}")  
            println("Max value is ${quotas.max}")  
        }  
    }  
}
```

- For API details, see [DescribeAccountAttributes](#) in *AWS SDK for Kotlin API reference*.

## Scenarios

### Get started with DB instances

The following code example shows how to:

- Create a custom DB parameter group and set parameter values.
- Create a DB instance that's configured to use the parameter group. The DB instance also contains a database.
- Take a snapshot of the instance.

- Delete the instance and parameter group.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
Before running this Java V2 code example, set up your development environment,  
including your credentials.  
  
For more information, see the following documentation topic:  
https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
  
This example performs the following tasks:  
  
1. Returns a list of the available DB engines by invoking the DescribeDbEngineVersions  
method.  
2. Selects an engine family and create a custom DB parameter group by invoking the  
createDBParameterGroup method.  
3. Gets the parameter groups by invoking the DescribeDbParameterGroups method.  
4. Gets parameters in the group by invoking the DescribeDbParameters method.  
5. Modifies both the auto_increment_offset and auto_increment_increments parameters by  
invoking the modifyDBParameterGroup method.  
6. Gets and displays the updated parameters.  
7. Gets a list of allowed engine versions by invoking the describeDbEngineVersions  
method.  
8. Gets a list of micro instance classes available for the selected engine.  
9. Creates an RDS database instance that contains a MySQL database and uses the  
parameter group  
10. Waits for DB instance to be ready and print out the connection endpoint value.  
11. Creates a snapshot of the DB instance.  
12. Waits for DB snapshot to be ready.  
13. Deletes the DB instance. rds.DeleteDbInstance.  
14. Deletes the parameter group.  
*/  
  
var sleepTime: Long = 20  
suspend fun main(args: Array<String>) {  
    val usage = """  
        Usage:  
        <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier> <dbName>  
        <masterUsername> <masterUserPassword> <dbSnapshotIdentifier>  
  
        Where:  
        dbGroupName - The database group name.  
        dbParameterGroupFamily - The database parameter group name.  
        dbInstanceIdentifier - The database instance identifier.  
        dbName - The database name.  
        username - The user name.  
        userPassword - The password that corresponds to the user name.  
        dbSnapshotIdentifier - The snapshot identifier.  
    """  
  
    if (args.size != 7) {  
        println(usage)  
        exitProcess(1)  
    }  
}
```

```
val dbGroupName = args[0]
val dbParameterGroupFamily = args[1]
val dbInstanceIdentifier = args[2]
val dbName = args[3]
val username = args[4]
val userPassword = args[5]
val dbSnapshotIdentifier = args[6]

println("1. Return a list of the available DB engines")
describeDBEngines()

println("2. Create a custom parameter group")
createDBParameterGroup(dbGroupName, dbParameterGroupFamily)

println("3. Get the parameter groups")
describeDbParameterGroups(dbGroupName)

println("4. Get the parameters in the group")
describeDbParameters(dbGroupName, 0)

println("5. Modify the auto_increment_offset parameter")
modifyDBParas(dbGroupName)

println("6. Display the updated value")
describeDbParameters(dbGroupName, -1)

println("7. Get a list of allowed engine versions")
getAllowedEngines(dbParameterGroupFamily)

println("8. Get a list of micro instance classes available for the selected
engine")
getMicroInstances()

println("9. Create an RDS database instance that contains a MySql database and uses
the parameter group")
val dbARN = createDatabaseInstance(dbGroupName, dbInstanceIdentifier, dbName,
username, userPassword)
println("The ARN of the new database is $dbARN")

println("10. Wait for DB instance to be ready")
waitForDbInstanceReady(dbInstanceIdentifier)

println("11. Create a snapshot of the DB instance")
createDbSnapshot(dbInstanceIdentifier, dbSnapshotIdentifier)

println("12. Wait for DB snapshot to be ready")
waitForSnapshotReady(dbInstanceIdentifier, dbSnapshotIdentifier)

println("13. Delete the DB instance")
deleteDbInstance(dbInstanceIdentifier)

println("14. Delete the parameter group")
if (dbARN != null) {
    deleteParaGroup(dbGroupName, dbARN)
}

println("The Scenario has successfully completed.")

}suspend fun deleteParaGroup(dbGroupName: String, dbARN: String) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
```

```

        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false // Reset this value.
            didFind = false // Reset this value.
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(dbARN) == 0) {
                        println("$dbARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the database
name.
                        isDataDel = true
                    }
                    index++
                }
            }
        }

        // Delete the para group.
        val parameterGroupRequest = DeleteDbParameterGroupRequest {
            dbParameterGroupName = dbGroupName
        }
        rdsClient.deleteDbParameterGroup(parameterGroupRequest)
        println("$dbGroupName was deleted.")
    }
}

suspend fun deleteDbInstance(dbIdentifierVal: String) {
    val deleteDbInstanceRequest = DeleteDbInstanceRequest {
        dbIdentifier = dbIdentifierVal
        deleteAutomatedBackups = true
        skipFinalSnapshot = true
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is ${response.dbInstance?.dbInstanceState}")
    }
}

// Waits until the snapshot instance is available.
suspend fun waitForSnapshotReady(dbIdentifierVal: String?,
                                dbSnapshotIdentifierVal: String?) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest = DescribeDbSnapshotsRequest {
        dbSnapshotIdentifier = dbSnapshotIdentifierVal
        dbIdentifier = dbIdentifierVal
    }

    while (!snapshotReady) {
        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbSnapshots(snapshotsRequest)
            val snapshotList: List<DbSnapshot>? = response.dbSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {

```

```
        snapshotReady = true
    } else {
        print(".")
        delay(sleepTime * 1000)
    }
}
}
}
println("The Snapshot is available!")
}

// Create an Amazon RDS snapshot.
suspend fun createDbSnapshot(dbInstanceIdentifierVal: String?, dbSnapshotIdentifierVal: String?) {
    val snapshotRequest = CreateDbSnapshotRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
        dbSnapshotIdentifier = dbSnapshotIdentifierVal
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbSnapshot(snapshotRequest)
        print("The Snapshot id is ${response.dbSnapshot?.dbiResourceId}")
    }
}

// Waits until the database instance is available.
suspend fun waitForDbInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest = DescribeDbInstancesRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
    }
    var endpoint = ""
    while (!instanceReady) {
        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.describeDbInstances(instanceRequest)
            val instanceList = response.dbInstances
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceReadyStr = instance.dbInstanceState.toString()
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint?.address.toString()
                        instanceReady = true
                    } else {
                        print(".")
                        delay(sleepTime * 1000)
                    }
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is $endpoint")
}

// Create a database instance and return the ARN of the database.
suspend fun createDatabaseInstance(dbGroupNameVal: String?, dbInstanceIdentifierVal: String?, dbNameVal: String?, masterUsernameVal: String?, masterUserPasswordVal: String?): String? {
    val instanceRequest = CreateDbInstanceRequest {
        dbInstanceIdentifier = dbInstanceIdentifierVal
        allocatedStorage = 100
        dbName = dbNameVal
        dbParameterGroupName = dbGroupNameVal
    }
}
```

```
        engine = "mysql"
        dbInstanceClass = "db.m4.large"
        engineVersion = "8.0"
        storageType = "standard"
        masterUsername = masterUsernameVal
        masterUserPassword = masterUserPasswordVal
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceState}")
        return response.dbInstance?.dbInstanceArn
    }
}

// Get a list of micro instances.
suspend fun getMicroInstances() {
    val dbInstanceOptionsRequest = DescribeOrderableDbInstanceOptionsRequest {
        engine = "mysql"
    }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response =
            rdsClient.describeOrderableDbInstanceOptions(dbInstanceOptionsRequest)
        val orderableDBInstances = response.orderableDbInstanceOptions
        if (orderableDBInstances != null) {
            for (dbInstanceOption in orderableDBInstances) {
                println("The engine version is ${dbInstanceOption.engineVersion}")
                println("The engine description is ${dbInstanceOption.engine}")
            }
        }
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest = DescribeDbEngineVersionsRequest {
        dbParameterGroupFamily = dbParameterGroupFamilyVal
        engine = "mysql"
    }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        val dbEngines: List<DbEngineVersion>? = response.dbEngineVersions
        if (dbEngines != null) {
            for (dbEngine in dbEngines) {
                println("The engine version is ${dbEngine.engineVersion}")
                println("The engine description is ${dbEngine.dbEngineDescription}")
            }
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBParas(dbGroupName: String) {
    val parameter1 = Parameter {
        parameterName = "auto_increment_offset"
        applyMethod = ApplyMethod.Immediate
        parameterValue = "5"
    }

    val paraList: ArrayList<Parameter> = ArrayList()
    paraList.add(parameter1)
    val groupRequest = ModifyDbParameterGroupRequest {
        dbParameterGroupName = dbGroupName
        parameters = paraList
    }
}
```

```

        RdsClient { region = "us-west-2" }.use { rdsClient ->
            val response = rdsClient.modifyDbParameterGroup(groupRequest)
            println("The parameter group ${response.dbParameterGroupName} was successfully
modified")
        }
    }

// Retrieve parameters in the group.
suspend fun describeDbParameters(dbGroupName: String?, flag: Int) {
    val dbParameterGroupsRequest: DescribeDbParametersRequest
    dbParameterGroupsRequest = if (flag == 0) {
        DescribeDbParametersRequest {
            dbParameterGroupName = dbGroupName
        }
    } else {
        DescribeDbParametersRequest {
            dbParameterGroupName = dbGroupName
            source = "user"
        }
    }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameters(dbParameterGroupsRequest)
        val dbParameters: List<Parameter>? = response.parameters
        var paraName: String
        if (dbParameters != null) {
            for (para in dbParameters) {
                // Only print out information about either auto_increment_offset or
auto_increment_increment.
                paraName = para.parameterName.toString()
                if (paraName.compareTo("auto_increment_offset") == 0 || paraName.compareTo("auto_increment_increment ") == 0) {
                    println("**** The parameter name is $paraName")
                    System.out.println("**** The parameter value is
${para.parameterValue}")
                    System.out.println("**** The parameter data type is
${para.dataType}")
                    System.out.println("**** The parameter description is
${para.description}")
                    System.out.println("**** The parameter allowed values is
${para.allowedValues}")
                }
            }
        }
    }
}

suspend fun describeDbParameterGroups(dbGroupName: String?) {
    val groupsRequest = DescribeDbParameterGroupsRequest {
        dbParameterGroupName = dbGroupName
        maxRecords = 20
    }
    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbParameterGroups(groupsRequest)
        val groups = response.dbParameterGroups
        if (groups != null) {
            for (group in groups) {
                println("The group name is ${group.dbParameterGroupName}")
                println("The group description is ${group.description}")
            }
        }
    }
}

// Create a parameter group.
suspend fun createDBParameterGroup(dbGroupName: String?, dbParameterGroupFamilyVal:
String?) {

```

```
    val groupRequest = CreateDbParameterGroupRequest {
        dbParameterGroupName = dbGroupName
        dbParameterGroupFamily = dbParameterGroupFamilyVal
        description = "Created by using the AWS SDK for Kotlin"
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbParameterGroup(groupRequest)
        println("The group name is ${response.dbParameterGroup?.dbParameterGroupName}")
    }
}

// Returns a list of the available DB engines.
suspend fun describeDBEngines() {
    val engineVersionsRequest = DescribeDbEngineVersionsRequest {
        defaultOnly = true
        engine = "mysql"
        maxRecords = 20
    }

    RdsClient { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        val engines: List<DbEngineVersion>? = response.dbEngineVersions

        // Get all DbEngineVersion objects.
        if (engines != null) {
            for (engineOb in engines) {
                println("The name of the DB parameter group family for the database
engine is ${engineOb.dbParameterGroupFamily}.")
                println("The name of the database engine ${engineOb.engine}.")
                println("The version number of the database engine
${engineOb.engineVersion}")
            }
        }
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.

- [CreateDBInstance](#)
- [CreateDBParameterGroup](#)
- [CreateDBSnapshot](#)
- [DeleteDBInstance](#)
- [DeleteDBParameterGroup](#)
- [DescribeDBEngineVersions](#)
- [DescribeDBInstances](#)
- [DescribeDBParameterGroups](#)
- [DescribeDBParameters](#)
- [DescribeDBSnapshots](#)
- [DescribeOrderableDBInstanceOptions](#)
- [ModifyDBParameterGroup](#)

## Amazon Redshift examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Redshift.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3610\)](#)

## Actions

### Create a cluster

The following code example shows how to create an Amazon Redshift cluster.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create the cluster.

```
suspend fun createCluster(clusterId: String?, masterUsernameVal: String?,  
    masterUserPasswordVal: String?) {  
    val clusterRequest = CreateClusterRequest {  
        clusterIdentifier = clusterId  
        masterUsername = masterUsernameVal // set the user name here  
        masterUserPassword = masterUserPasswordVal // set the user password here  
        nodeType = "ds2.xlarge"  
        publiclyAccessible = true  
        numberofNodes = 2  
    }  
  
    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->  
        val clusterResponse = redshiftClient.createCluster(clusterRequest)  
        println("Created cluster ${clusterResponse.cluster?.clusterIdentifier}")  
    }  
}
```

- For API details, see [CreateCluster](#) in *AWS SDK for Kotlin API reference*.

### Delete a cluster

The following code example shows how to delete an Amazon Redshift cluster.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete the cluster.

```
suspend fun deleteRedshiftCluster(clusterId: String?) {
```

```
    val request = DeleteClusterRequest {
        clusterIdentifier = clusterId
        skipFinalClusterSnapshot = true
    }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val response = redshiftClient.deleteCluster(request)
        println("The status is ${response.cluster?.clusterStatus}")
    }
}
```

- For API details, see [DeleteCluster](#) in *AWS SDK for Kotlin API reference*.

## Describe your clusters

The following code example shows how to describe your Amazon Redshift clusters.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Describe the cluster.

```
suspend fun describeRedshiftClusters() {

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.describeClusters(DescribeClustersRequest {})
        val clusterList = clusterResponse.clusters

        if (clusterList != null) {
            for (cluster in clusterList) {
                println("Cluster database name is ${cluster.dbName}")
                println("Cluster status is ${cluster.clusterStatus}")
            }
        }
    }
}
```

- For API details, see [DescribeClusters](#) in *AWS SDK for Kotlin API reference*.

## Modify a cluster

The following code example shows how to modify an Amazon Redshift cluster.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Modify a cluster.

```
suspend fun modifyCluster(clusterId: String?) {
    val modifyClusterRequest = ModifyClusterRequest {
        clusterIdentifier = clusterId
        preferredMaintenanceWindow = "wed:07:30-wed:08:00"
    }

    RedshiftClient { region = "us-west-2" }.use { redshiftClient ->
        val clusterResponse = redshiftClient.modifyCluster(modifyClusterRequest)
        println("The modified cluster was successfully modified and has
        ${clusterResponse.cluster?.preferredMaintenanceWindow} as the maintenance window")
    }
}
```

- For API details, see [ModifyCluster](#) in *AWS SDK for Kotlin API reference*.

## Amazon Rekognition examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Rekognition.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3612\)](#)
- [Scenarios \(p. 3620\)](#)

## Actions

### Compare faces in an image against a reference image

The following code example shows how to compare faces in an image against a reference image with Amazon Rekognition.

For more information, see [Comparing faces in images](#).

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun compareTwoFaces(similarityThresholdVal: Float, sourceImageVal: String,
targetImageVal: String) {

    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage = Image {
        bytes = sourceBytes
    }
}
```

```
    val tarImage = Image {
        bytes = targetBytes
    }

    val facesRequest = CompareFacesRequest {
        sourceImage = souImage
        targetImage = tarImage
        similarityThreshold = similarityThresholdVal
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->

        val compareFacesResult = rekClient.compareFaces(facesRequest)
        val faceDetails = compareFacesResult.faceMatches

        if (faceDetails != null) {
            for (match: CompareFacesMatch in faceDetails) {
                val face = match.face
                val position = face?.boundingBox
                if (position != null)
                    println("Face at ${position.left} ${position.top} matches with
${face.confidence} % confidence.")
            }
        }

        val uncompered = compareFacesResult.unmatchedFaces
        if (uncompered != null)
            println("There was ${uncompered.size} face(s) that did not match")

        println("Source image rotation:
${compareFacesResult.sourceImageOrientationCorrection}")
        println("target image rotation:
${compareFacesResult.targetImageOrientationCorrection}")
    }
}
```

- For API details, see [CompareFaces](#) in *AWS SDK for Kotlin API reference*.

## Create a collection

The following code example shows how to create an Amazon Rekognition collection.

For more information, see [Creating a collection](#).

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createMyCollection(collectionIdVal: String) {

    val request = CreateCollectionRequest {
        collectionId = collectionIdVal
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
```

```
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- For API details, see [CreateCollection in AWS SDK for Kotlin API reference](#).

## Delete a collection

The following code example shows how to delete an Amazon Rekognition collection.

For more information, see [Deleting a collection](#).

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteMyCollection(collectionIdVal: String) {

    val request = DeleteCollectionRequest {
        collectionId = collectionIdVal
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- For API details, see [DeleteCollection in AWS SDK for Kotlin API reference](#).

## Delete faces from a collection

The following code example shows how to delete faces from an Amazon Rekognition collection.

For more information, see [Deleting faces from a collection](#).

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteFacesCollection(collectionIdVal: String?, faceIdVal: String) {

    val deleteFacesRequest = DeleteFacesRequest {
        collectionId = collectionIdVal
        faceIds = listOf(faceIdVal)
    }
}
```

```
        RekognitionClient { region = "us-east-1" }.use { rekClient ->
            rekClient.deleteFaces(deleteFacesRequest)
            println("$faceIdVal was deleted from the collection")
        }
    }
```

- For API details, see [DeleteFaces](#) in *AWS SDK for Kotlin API reference*.

## Describe a collection

The following code example shows how to describe an Amazon Rekognition collection.

For more information, see [Describing a collection](#).

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeColl(collectionName: String) {

    val request = DescribeCollectionRequest {
        collectionId = collectionName
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

- For API details, see [DescribeCollection](#) in *AWS SDK for Kotlin API reference*.

## Detect faces in an image

The following code example shows how to detect faces in an image with Amazon Rekognition.

For more information, see [Detecting faces in an image](#).

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun detectFacesinImage(sourceImage: String?) {

    val souImage = Image {
        bytes = (File(sourceImage)).readBytes()
```

```
    }

    val request = DetectFacesRequest {
        attributes = listOf(Attribute.All)
        image = souImage
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between ${ageRange?.low} and
${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}
```

- For API details, see [DetectFaces](#) in *AWS SDK for Kotlin API reference*.

## Detect labels in an image

The following code example shows how to detect labels in an image with Amazon Rekognition.

For more information, see [Detecting labels in an image](#).

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun detectImageLabels(sourceImage: String) {

    val souImage = Image {
        bytes = (File(sourceImage)).readBytes()
    }
    val request = DetectLabelsRequest {
        image = souImage
        maxLabels = 10
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- For API details, see [DetectLabels](#) in *AWS SDK for Kotlin API reference*.

## Detect moderation labels in an image

The following code example shows how to detect moderation labels in an image with Amazon Rekognition. Moderation labels identify content that may be inappropriate for some audiences.

For more information, see [Detecting inappropriate images](#).

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun detectModLabels(sourceImage: String) {

    val myImage = Image {
        this.bytes = (File(sourceImage).readBytes())
    }

    val request = DetectModerationLabelsRequest {
        image = myImage
        minConfidence = 60f
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} % Parent: ${label.parentName}")
        }
    }
}
```

- For API details, see [DetectModerationLabels](#) in *AWS SDK for Kotlin API reference*.

### Detect text in an image

The following code example shows how to detect text in an image with Amazon Rekognition.

For more information, see [Detecting text in an image](#).

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun detectTextLabels(sourceImage: String?) {

    val souImage = Image {
        bytes = (File(sourceImage).readBytes())
    }

    val request = DetectTextRequest {
        image = souImage
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
    }
}
```

```
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id: ${text.parentId}")
            println("Type: ${text.type}")
        }
    }
}
```

- For API details, see [DetectText](#) in *AWS SDK for Kotlin API reference*.

## Index faces to a collection

The following code example shows how to index faces in an image and add them to an Amazon Rekognition collection.

For more information, see [Adding faces to a collection](#).

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun addToCollection(collectionIdVal: String?, sourceImage: String) {

    val souImage = Image {
        bytes = (File(sourceImage).readBytes())
    }

    val request = IndexFacesRequest {
        collectionId = collectionIdVal
        image = souImage
        maxFaces = 1
        qualityFilter = QualityFilter.Auto
        detectionAttributes = listOf(Attribute.Default)
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)

        // Display the results.
        println("Results for the image")
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
            println("Face ID: ${faceRecord.face?.faceId}")
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
            println("Reasons:")

            unindexedFace.reasons?.forEach { reason ->
                println("Reason: $reason")
            }
        }
    }
}
```

```
        }  
    }  
}
```

- For API details, see [IndexFaces](#) in *AWS SDK for Kotlin API reference*.

## List collections

The following code example shows how to list Amazon Rekognition collections.

For more information, see [Listing collections](#).

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAllCollections() {  
  
    val request = ListCollectionsRequest {  
        maxResults = 10  
    }  
  
    RekognitionClient { region = "us-east-1" }.use { rekClient ->  
        val response = rekClient.listCollections(request)  
        response.collectionIds?.forEach { resultId ->  
            println(resultId)  
        }  
    }  
}
```

- For API details, see [ListCollections](#) in *AWS SDK for Kotlin API reference*.

## List faces in a collection

The following code example shows how to list faces in an Amazon Rekognition collection.

For more information, see [Listing faces in a collection](#).

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listFacesCollection(collectionIdVal: String?) {  
  
    val request = ListFacesRequest {  
        collectionId = collectionIdVal  
        maxResults = 10  
    }
```

```
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listFaces(request)
        response.faces?.forEach { face ->
            println("Confidence level there is a face: ${face.confidence}")
            println("The face Id value is ${face.faceId}")
        }
    }
}
```

- For API details, see [ListFaces](#) in *AWS SDK for Kotlin API reference*.

## Recognize celebrities in an image

The following code example shows how to recognize celebrities in an image with Amazon Rekognition.

For more information, see [Recognizing celebrities in an image](#).

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {

    val souImage = Image {
        bytes = (File(sourceImage).readBytes())
    }

    val request = RecognizeCelebritiesRequest {
        image = souImage
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
        println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
    }
}
```

- For API details, see [RecognizeCelebrities](#) in *AWS SDK for Kotlin API reference*.

## Scenarios

### Detect information in videos

The following code example shows how to:

- Start Amazon Rekognition jobs to detect elements like people, objects, and text in videos.
- Check job status until jobs finish.
- Output the list of elements detected by each job.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Detect faces in a video stored in an Amazon S3 bucket.

```
suspend fun startFaceDetection(channelVal: NotificationChannel?, bucketVal: String, videoVal: String) {

    val s3Obj = S3Object {
        bucket = bucketVal
        name = videoVal
    }
    val vidOb = Video {
        s3Object = s3Obj
    }

    val request = StartFaceDetectionRequest {
        jobTag = "Faces"
        faceAttributes = FaceAttributes.All
        notificationChannel = channelVal
        video = vidOb
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val startLabelDetectionResult = rekClient.startFaceDetection(request)
        startJobId = startLabelDetectionResult.jobId.toString()
    }
}

suspend fun getFaceResults() {

    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var response: GetFaceDetectionResponse? = null

        val recognitionRequest = GetFaceDetectionRequest {
            jobId = startJobId
            maxResults = 10
        }

        // Wait until the job succeeds.
        while (!finished) {
            response = rekClient.getFaceDetection(recognitionRequest)
            status = response.jobStatus.toString()
            if (status.compareTo("SUCCEEDED") == 0)
                finished = true
            else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }
    }
}
```

```

    }

    // Proceed when the job is done - otherwise VideoMetadata is null.
    val videoMetaData = response?.videoMetadata
    println("Format: ${videoMetaData?.format}")
    println("Codec: ${videoMetaData?.codec}")
    println("Duration: ${videoMetaData?.durationMillis}")
    println("FrameRate: ${videoMetaData?.frameRate}")

    // Show face information.
    response?.faces?.forEach { face ->
        println("Age: ${face.face?.ageRange}")
        println("Face: ${face.face?.beard}")
        println("Eye glasses: ${face?.face?.eyeglasses}")
        println("Mustache: ${face.face?.mustache}")
        println("Smile: ${face.face?.smile}")
    }
}

}

```

Detect inappropriate or offensive content in a video stored in an Amazon S3 bucket.

```

suspend fun startModerationDetection(channel: NotificationChannel?, bucketVal: String?,
                                     videoVal: String?) {

    val s3Obj = S3Object {
        bucket = bucketVal
        name = videoVal
    }
    val vidOb = Video {
        s3Object = s3Obj
    }
    val request = StartContentModerationRequest {
        jobTag = "Moderation"
        notificationChannel = channel
        video = vidOb
    }

    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        val startModDetectionResult = rekClient.startContentModeration(request)
        startJobId = startModDetectionResult.jobId.toString()
    }
}

suspend fun getModResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var modDetectionResponse: GetContentModerationResponse? = null

        val modRequest = GetContentModerationRequest {
            jobId = startJobId
            maxResults = 10
        }

        // Wait until the job succeeds.
        while (!finished) {
            modDetectionResponse = rekClient.getContentModeration(modRequest)
            status = modDetectionResponse.jobStatus.toString()
            if (status.compareTo("SUCCEEDED") == 0)
                finished = true
            else {
                println("$yy status is: $status")
            }
        }
    }
}

```

```
        delay(1000)
    }
    yy++
}

// Proceed when the job is done - otherwise VideoMetadata is null.
val videoMetaData = modDetectionResponse?.videoMetadata
println("Format: ${videoMetaData?.format}")
println("Codec: ${videoMetaData?.codec}")
println("Duration: ${videoMetaData?.durationMillis}")
println("FrameRate: ${videoMetaData?.frameRate}")

modDetectionResponse?.moderationLabels?.forEach { mod ->
    val seconds: Long = mod.timestamp / 1000
    print("Mod label: $seconds ")
    println(mod.moderationLabel)
}
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.

- [GetCelebrityRecognition](#)
- [GetContentModeration](#)
- [GetLabelDetection](#)
- [GetPersonTracking](#)
- [GetSegmentDetection](#)
- [GetTextDetection](#)
- [StartCelebrityRecognition](#)
- [StartContentModeration](#)
- [StartLabelDetection](#)
- [StartPersonTracking](#)
- [StartSegmentDetection](#)
- [StartTextDetection](#)

## Amazon S3 examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Simple Storage Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3623\)](#)
- [Scenarios \(p. 3629\)](#)

## Actions

### Copy an object from one bucket to another

The following code example shows how to copy an S3 object from one bucket to another.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun copyBucketObject(  
    fromBucket: String,  
    objectKey: String,  
    toBucket: String  
) {  
  
    var encodedUrl = ""  
    try {  
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",  
StandardCharsets.UTF_8.toString())  
    } catch (e: UnsupportedEncodingException) {  
        println("URL could not be encoded: " + e.message)  
    }  
  
    val request = CopyObjectRequest {  
        copySource = encodedUrl  
        bucket = toBucket  
        key = objectKey  
    }  
    S3Client { region = "us-east-1" }.use { s3 ->  
        s3.copyObject(request)  
    }  
}
```

- For API details, see [CopyObject](#) in *AWS SDK for Kotlin API reference*.

## Create a bucket

The following code example shows how to create an S3 bucket.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createNewBucket(bucketName: String) {  
  
    val request = CreateBucketRequest {  
        bucket = bucketName  
    }  
  
    S3Client { region = "us-east-1" }.use { s3 ->  
        s3.createBucket(request)  
        println("$bucketName is ready")  
    }  
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for Kotlin API reference*.

## Delete a policy from a bucket

The following code example shows how to delete a policy from an S3 bucket.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {  
  
    val request = DeleteBucketPolicyRequest {  
        bucket = bucketName  
    }  
  
    S3Client { region = "us-east-1" }.use { s3 ->  
        s3.deleteBucketPolicy(request)  
        println("Done!")  
    }  
}
```

- For API details, see [DeleteBucketPolicy](#) in *AWS SDK for Kotlin API reference*.

## Delete multiple objects

The following code example shows how to delete multiple objects from an S3 bucket.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteBucketObjects(bucketName: String, objectName: String) {  
  
    val objectId = ObjectIdentifier {  
        key = objectName  
    }  
  
    val delOb = Delete {  
        objects = listOf(objectId)  
    }  
  
    val request = DeleteObjectsRequest {  
        bucket = bucketName  
        delete = delOb  
    }  
  
    S3Client { region = "us-east-1" }.use { s3 ->  
        s3.deleteObjects(request)  
    }  
}
```

```
        println("$objectName was deleted from $bucketName")
    }
}
```

- For API details, see [DeleteObjects](#) in *AWS SDK for Kotlin API reference*.

## Get an object from a bucket

The following code example shows how to read data from an object in an S3 bucket.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getObjectBytes(bucketName: String, keyName: String, path: String) {

    val request = GetObjectRequest {
        key = keyName
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}
```

- For API details, see [GetObject](#) in *AWS SDK for Kotlin API reference*.

## Get the ACL of an object

The following code example shows how to get the access control list (ACL) of an S3 object.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getBucketACL(objectKey: String, bucketName: String) {

    val request = GetObjectAclRequest {
        bucket = bucketName
        key = objectKey
    }

    S3Client { region = "us-east-1" }.use { s3 ->
```

```
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- For API details, see [GetObjectAcl](#) in *AWS SDK for Kotlin API reference*.

## Get the policy for a bucket

The following code example shows how to get the policy for an S3 bucket.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")

    val request = GetBucketPolicyRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        val policyRes = s3.getBucketPolicy(request)
        return policyRes.policy
    }
}
```

- For API details, see [GetBucketPolicy](#) in *AWS SDK for Kotlin API reference*.

## List objects in a bucket

The following code example shows how to list objects in an S3 bucket.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listBucketObjects(bucketName: String) {
    val request = ListObjectsRequest {
        bucket = bucketName
    }

    S3Client { region = "us-east-1" }.use { s3 ->
```

```
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${calKb(myObject.size)} KBs")
            println("The owner is ${myObject.owner}")
        }
    }

private fun calKb(intValue: Long): Long {
    return intValue / 1024
}
```

- For API details, see [ListObjects](#) in *AWS SDK for Kotlin API reference*.

## Set a new ACL for a bucket

The following code example shows how to set a new access control list (ACL) for an S3 bucket.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun setBucketAcl(bucketName: String, idVal: String) {

    val myGrant = Grantee {
        id = idVal
        type = Type.CanonicalUser
    }

    val ownerGrant = Grant {
        grantee = myGrant
        permission = Permission.FullControl
    }

    val grantList = mutableListOf<Grant>()
    grantList.add(ownerGrant)

    val ownerOb = Owner {
        id = idVal
    }

    val acl = AccessControlPolicy {
        owner = ownerOb
        grants = grantList
    }

    val request = PutBucketAclRequest {
        bucket = bucketName
        accessControlPolicy = acl
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

```
}
```

- For API details, see [PutBucketAcl](#) in *AWS SDK for Kotlin API reference*.

## Upload an object to a bucket

The following code example shows how to upload an object to an S3 bucket.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun putS3Object(bucketName: String, objectKey: String, objectPath: String) {  
  
    val metadataVal = mutableMapOf<String, String>()  
    metadataVal["myVal"] = "test"  
  
    val request = PutObjectRequest {  
        bucket = bucketName  
        key = objectKey  
        metadata = metadataVal  
        body = File(objectPath).asByteStream()  
    }  
  
    S3Client { region = "us-east-1" }.use { s3 ->  
        val response = s3.putObject(request)  
        println("Tag information is ${response.eTag}")  
    }  
}
```

- For API details, see [PutObject](#) in *AWS SDK for Kotlin API reference*.

## Scenarios

### Get started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.
- Delete a bucket.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun main(args: Array<String>) {  
  
    val usage = """  
Usage:  
    <bucketName> <key> <objectPath> <savePath> <toBucket>  
  
Where:  
    bucketName - The Amazon S3 bucket to create.  
    key - The key to use.  
    objectPath - The path where the file is located (for example, C:/AWS/  
book2.pdf).  
    savePath - The path where the file is saved after it's downloaded (for example,  
C:/AWS/book2.pdf).  
    toBucket - An Amazon S3 bucket to where an object is copied to (for example,  
C:/AWS/book2.pdf).  
    """  
  
    if (args.size != 4) {  
        println(usage)  
        exitProcess(1)  
    }  
  
    val bucketName = args[0]  
    val key = args[1]  
    val objectPath = args[2]  
    val savePath = args[3]  
    val toBucket = args[4]  
  
    // Create an Amazon S3 bucket.  
    createBucket(bucketName)  
  
    // Update a local file to the Amazon S3 bucket.  
    putObject(bucketName, key, objectPath)  
  
    // Download the object to another local file.  
    getObject(bucketName, key, savePath)  
  
    // List all objects located in the Amazon S3 bucket.  
    listBucketObs(bucketName)  
  
    // Copy the object to another Amazon S3 bucket  
    copyBucketOb(bucketName, key, toBucket)  
  
    // Delete the object from the Amazon S3 bucket.  
    deleteBucketObs(bucketName, key)  
  
    // Delete the Amazon S3 bucket.  
    deleteBucket(bucketName)  
    println("All Amazon S3 operations were successfully performed")  
}  
  
suspend fun createBucket(bucketName: String) {  
  
    val request = CreateBucketRequest {  
        bucket = bucketName  
    }  
  
    S3Client { region = "us-east-1" }.use { s3 ->  
        s3.createBucket(request)  
        println("$bucketName is ready")  
    }  
}
```

```
        }

    suspend fun putObject(bucketName: String, objectKey: String, objectPath: String) {

        val metadataVal = mutableMapOf<String, String>()
        metadataVal["myVal"] = "test"

        val request = PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }

        S3Client { region = "us-east-1" }.use { s3 ->
            val response = s3.putObject(request)
            println("Tag information is ${response.eTag}")
        }
    }

    suspend fun getObject(bucketName: String, keyName: String, path: String) {

        val request = GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

        S3Client { region = "us-east-1" }.use { s3 ->
            s3.getObject(request) { resp ->
                val myFile = File(path)
                resp.body?.writeToFile(myFile)
                println("Successfully read $keyName from $bucketName")
            }
        }
    }

    suspend fun listBucketObs(bucketName: String) {

        val request = ListObjectsRequest {
            bucket = bucketName
        }

        S3Client { region = "us-east-1" }.use { s3 ->

            val response = s3.listObjects(request)
            response.contents?.forEach { myObject ->
                println("The name of the key is ${myObject.key}")
                println("The owner is ${myObject.owner}")
            }
        }
    }

    suspend fun copyBucketOb(fromBucket: String, objectKey: String, toBucket: String) {

        var encodedUrl = ""
        try {
            encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
                StandardCharsets.UTF_8.toString())
        } catch (e: UnsupportedEncodingException) {
            println("URL could not be encoded: " + e.message)
        }

        val request = CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
        }
    }
}
```

```
        key = objectKey
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucket0bs(bucketName: String, objectName: String) {

    val objectId = ObjectIdentifier {
        key = objectName
    }

    val delOb = Delete {
        objects = listOf(objectId)
    }

    val request = DeleteObjectsRequest {
        bucket = bucketName
        delete = delOb
    }

    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {

    val request = DeleteBucketRequest {
        bucket = bucketName
    }
    S3Client { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

## Amazon SNS examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Simple Notification Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3633\)](#)

## Actions

### Add tags to a topic

The following code example shows how to add tags to an Amazon SNS topic.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun addTopicTags(topicArn: String) {  
  
    val tag = Tag {  
        key = "Team"  
        value = "Development"  
    }  
  
    val tag2 = Tag {  
        key = "Environment"  
        value = "Gamma"  
    }  
  
    val tagList = mutableListOf<Tag>()  
    tagList.add(tag)  
    tagList.add(tag2)  
  
    val request = TagResourceRequest {  
        resourceArn = topicArn  
        tags = tagList  
    }  
  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        snsClient.tagResource(request)  
        println("Tags have been added to $topicArn")  
    }  
}
```

- For API details, see [TagResource](#) in *AWS SDK for Kotlin API reference*.

### Create a topic

The following code example shows how to create an Amazon SNS topic.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createSNSTopic(topicName: String): String {  
  
    val request = CreateTopicRequest {  
        name = topicName  
    }  
  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        val result = snsClient.createTopic(request)  
        return result.topicArn.toString()  
    }  
}
```

- For API details, see [CreateTopic](#) in *AWS SDK for Kotlin API reference*.

## Delete a subscription

The following code example shows how to delete an Amazon SNS subscription.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun unSub(subscriptionArnVal: String) {  
  
    val request = UnsubscribeRequest {  
        subscriptionArn = subscriptionArnVal  
    }  
  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        snsClient.unsubscribe(request)  
        println("Subscription was removed for ${request.subscriptionArn}")  
    }  
}
```

- For API details, see [Unsubscribe](#) in *AWS SDK for Kotlin API reference*.

## Delete a topic

The following code example shows how to delete an Amazon SNS topic and all subscriptions to that topic.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteSNSTopic(topicArnVal: String) {
```

```
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- For API details, see [DeleteTopic](#) in *AWS SDK for Kotlin API reference*.

## Get the properties of a topic

The following code example shows how to get the properties of an Amazon SNS topic.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {

    val request = GetTopicAttributesRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for Kotlin API reference*.

## List the subscribers of a topic

The following code example shows how to retrieve the list of subscribers of an Amazon SNS topic.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listSNSSubscriptions() {

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
    }
}
```

```
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- For API details, see [ListSubscriptions](#) in *AWS SDK for Kotlin API reference*.

## List topics

The following code example shows how to list Amazon SNS topics.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listSNSTopics() {

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- For API details, see [ListTopics](#) in *AWS SDK for Kotlin API reference*.

## Publish an SMS text message

The following code example shows how to publish SMS messages using Amazon SNS.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun pubTextSMS(messageVal: String?, phoneNumberVal: String?) {

    val request = PublishRequest {
        message = messageVal
        phoneNumber = phoneNumberVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

```
    }
```

- For API details, see [Publish](#) in *AWS SDK for Kotlin API reference*.

## [Publish to a topic](#)

The following code example shows how to publish messages to an Amazon SNS topic.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun pubTopic(topicArnVal: String, messageVal: String) {  
  
    val request = PublishRequest {  
        message = messageVal  
        topicArn = topicArnVal  
    }  
  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        val result = snsClient.publish(request)  
        println("${result.messageId} message sent.")  
    }  
}
```

- For API details, see [Publish](#) in *AWS SDK for Kotlin API reference*.

## [Set topic attributes](#)

The following code example shows how to set Amazon SNS topic attributes.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun setTopAttr(attribute: String?, topicArnVal: String?, value: String?) {  
  
    val request = SetTopicAttributesRequest {  
        attributeName = attribute  
        attributeValue = value  
        topicArn = topicArnVal  
    }  
  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        snsClient.setTopicAttributes(request)  
        println("Topic ${request.topicArn} was updated.")  
    }  
}
```

```
    }
```

- For API details, see [SetTopicAttributes](#) in *AWS SDK for Kotlin API reference*.

## Subscribe a Lambda function to a topic

The following code example shows how to subscribe a Lambda function so it receives notifications from an Amazon SNS topic.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun subLambda(topicArnVal: String?, lambdaArn: String?) {  
  
    val request = SubscribeRequest {  
        protocol = "lambda"  
        endpoint = lambdaArn  
        returnSubscriptionArn = true  
        topicArn = topicArnVal  
    }  
  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        val result = snsClient.subscribe(request)  
        println(" The subscription Arn is ${result.subscriptionArn}")  
    }  
}
```

- For API details, see [Subscribe](#) in *AWS SDK for Kotlin API reference*.

## Subscribe an email address to a topic

The following code example shows how to subscribe an email address to an Amazon SNS topic.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun subEmail(topicArnVal: String, email: String): String {  
  
    val request = SubscribeRequest {  
        protocol = "email"  
        endpoint = email  
        returnSubscriptionArn = true  
        topicArn = topicArnVal  
    }
```

```
SnsClient { region = "us-east-1" }.use { snsClient ->
    val result = snsClient.subscribe(request)
    return result.subscriptionArn.toString()
}
}
```

- For API details, see [Subscribe](#) in *AWS SDK for Kotlin API reference*.

## Amazon SQS examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with Amazon Simple Queue Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3639\)](#)

## Actions

### Create a queue

The following code example shows how to create an Amazon SQS queue.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createQueue(queueNameVal: String): String {
    println("Create Queue")
    val createQueueRequest = CreateQueueRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.createQueue(createQueueRequest)
        println("Get queue url")

        val getQueueUrlRequest = GetQueueUrlRequest {
            queueName = queueNameVal
        }

        val getQueueUrlResponse = sqsClient.getQueueUrl(getQueueUrlRequest)
        return getQueueUrlResponse.queueUrl.toString()
    }
}
```

- For API details, see [CreateQueue](#) in *AWS SDK for Kotlin API reference*.

## Delete a message from a queue

The following code example shows how to delete a message from an Amazon SQS queue.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteMessages(queueUrlVal: String) {  
    println("Delete Messages from $queueUrlVal")  
  
    val purgeRequest = PurgeQueueRequest {  
        queueUrl = queueUrlVal  
    }  
  
    SqsClient { region = "us-east-1" }.use { sqsClient ->  
        sqsClient.purgeQueue(purgeRequest)  
        println("Messages are successfully deleted from $queueUrlVal")  
    }  
}  
  
suspend fun deleteQueue(queueUrlVal: String) {  
  
    val request = DeleteQueueRequest {  
        queueUrl = queueUrlVal  
    }  
  
    SqsClient { region = "us-east-1" }.use { sqsClient ->  
        sqsClient.deleteQueue(request)  
        println("$queueUrlVal was deleted!")  
    }  
}
```

- For API details, see [DeleteMessage](#) in *AWS SDK for Kotlin API reference*.

## Delete a queue

The following code example shows how to delete an Amazon SQS queue.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteMessages(queueUrlVal: String) {  
    println("Delete Messages from $queueUrlVal")
```

```
    val purgeRequest = PurgeQueueRequest {
        queueUrl = queueUrlVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.purgeQueue(purgeRequest)
        println("Messages are successfully deleted from $queueUrlVal")
    }
}

suspend fun deleteQueue(queueUrlVal: String) {

    val request = DeleteQueueRequest {
        queueUrl = queueUrlVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteQueue(request)
        println("$queueUrlVal was deleted!")
    }
}
```

- For API details, see [DeleteQueue](#) in *AWS SDK for Kotlin API reference*.

## List queues

The following code example shows how to list Amazon SQS queues.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listQueues() {
    println("\nList Queues")

    val prefix = "que"
    val listQueuesRequest = ListQueuesRequest {
        queueNamePrefix = prefix
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.listQueues(listQueuesRequest)
        response.queueUrls?.forEach { url ->
            println(url)
        }
    }
}
```

- For API details, see [ListQueues](#) in *AWS SDK for Kotlin API reference*.

## Receive messages from a queue

The following code example shows how to receive messages from an Amazon SQS queue.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun receiveMessages(queueUrlVal: String?) {  
  
    println("Retrieving messages from $queueUrlVal")  
  
    val receiveMessageRequest = ReceiveMessageRequest {  
        queueUrl = queueUrlVal  
        maxNumberOfMessages = 5  
    }  
  
    SqsClient { region = "us-east-1" }.use { sqsClient ->  
        val response = sqsClient.receiveMessage(receiveMessageRequest)  
        response.messages?.forEach { message ->  
            println(message.body)  
        }  
    }  
}
```

- For API details, see [ReceiveMessage in AWS SDK for Kotlin API reference](#).

## Send a message to a queue

The following code example shows how to send a message to an Amazon SQS queue.

## SDK for Kotlin

### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun sendMessages(queueUrlVal: String, message: String) {  
    println("Sending multiple messages")  
    println("\nSend message")  
    val sendRequest = SendMessageRequest {  
        queueUrl = queueUrlVal  
        messageBody = message  
        delaySeconds = 10  
    }  
  
    SqsClient { region = "us-east-1" }.use { sqsClient ->  
        sqsClient.sendMessage(sendRequest)  
        println("A single message was successfully sent.")  
    }  
}  
  
suspend fun sendBatchMessages(queueUrlVal: String?) {  
    println("Sending multiple messages")  
  
    val msg1 = SendMessageBatchRequestEntry {
```

```
        id = "id1"
        messageBody = "Hello from msg 1"
    }

    val msg2 = SendMessageBatchRequestEntry {
        id = "id2"
        messageBody = "Hello from msg 2"
    }

    val sendMessageBatchRequest = SendMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = listOf(msg1, msg2)
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.sendMessageBatch(sendMessageBatchRequest)
        println("Batch message were successfully sent.")
    }
}
```

- For API details, see [SendMessage](#) in *AWS SDK for Kotlin API reference*.

## Secrets Manager examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with AWS Secrets Manager.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3643\)](#)

## Actions

### Create a secret

The following code example shows how to create a Secrets Manager secret.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createNewSecret(secretName: String?, secretValue: String?): String? {

    val request = CreateSecretRequest {
        name = secretName
        description = "This secret was created by the AWS Secrets Manager Kotlin API"
        secretString = secretValue
    }
}
```

```
    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        val response = secretsClient.createSecret(request)
        return response.arn
    }
}
```

- For API details, see [CreateSecret](#) in *AWS SDK for Kotlin API reference*.

## Describe a secret

The following code example shows how to describe a Secrets Manager secret.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeGivenSecret(secretName: String?) {

    val secretRequest = DescribeSecretRequest {
        secretId = secretName
    }

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        val response = secretsClient.describeSecret(secretRequest)
        val secArn = response.description
        println("The secret description is $secArn")
    }
}
```

- For API details, see [DescribeSecret](#) in *AWS SDK for Kotlin API reference*.

## Get a secret value

The following code example shows how to get a Secrets Manager secret value.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getValue(secretName: String?) {

    val valueRequest = GetSecretValueRequest {
        secretId = secretName
    }

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
```

```
        val response = secretsClient.getSecretValue(valueRequest)
        val secret = response.secretString
        println("The secret value is $secret")
    }
}
```

- For API details, see [GetSecretValue](#) in *AWS SDK for Kotlin API reference*.

## List secrets

The following code example shows how to list Secrets Manager secrets.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listAllSecrets() {

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        val response = secretsClient.listSecrets(ListSecretsRequest {})
        response.secretList?.forEach { secret ->
            println("The secret name is ${secret.name}")
            println("The secret description is ${secret.description}")
        }
    }
}
```

- For API details, see [ListSecrets](#) in *AWS SDK for Kotlin API reference*.

## Put a value in a secret

The following code example shows how to put a value in a Secrets Manager secret.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun updateMySecret(secretName: String?, secretValue: String?) {

    val request = UpdateSecretRequest {
        secretId = secretName
        secretString = secretValue
    }

    SecretsManagerClient { region = "us-east-1" }.use { secretsClient ->
        secretsClient.updateSecret(request)
    }
}
```

```
        println("The secret value was updated")
    }
}
```

- For API details, see [PutSecretValue](#) in *AWS SDK for Kotlin API reference*.

## Step Functions examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with AWS Step Functions.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3646\)](#)

## Actions

### Create a state machine

The following code example shows how to create a Step Functions state machine.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static String createMachine( SfnClient sfnClient, String roleARN, String
stateMachineName, String jsonFile) {

    String json = getJSONString(jsonFile);
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
            .definition(json)
            .name(stateMachineName)
            .roleArn(roleARN)
            .type(StateMachineType.STANDARD)
            .build();

        CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
        return response.stateMachineArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

```
private static String getJSONString(String path) {  
    try {  
        JSONParser parser = new JSONParser();  
        JSONObject data = (JSONObject) parser.parse(new FileReader(path)); //path to  
        the JSON file.  
        return data.toJSONString();  
  
    } catch (IOException | org.json.simple.parser.ParseException e) {  
        e.printStackTrace();  
    }  
    return "";  
}
```

- For API details, see [CreateStateMachine](#) in *AWS SDK for Kotlin API reference*.

## Delete a state machine

The following code example shows how to delete a Step Functions state machine.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun deleteMachine(stateMachineArnVal: String) {  
    val deleteStateMachineRequest = DeleteStateMachineRequest {  
        stateMachineArn = stateMachineArnVal  
    }  
  
    SfnClient { region = "us-east-1" }.use { sfnClient ->  
        sfnClient.deleteStateMachine(deleteStateMachineRequest)  
        println("$stateMachineArnVal was successfully deleted.")  
    }  
}
```

- For API details, see [DeleteStateMachine](#) in *AWS SDK for Kotlin API reference*.

## List state machine runs

The following code example shows how to list Step Functions state machine runs.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getExeHistory(exeARN: String?) {
```

```
    val historyRequest = GetExecutionHistoryRequest {
        executionArn = exeARN
        maxResults = 10
    }

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.getExecutionHistory(historyRequest)
        response.events?.forEach { event ->
            println("The event type is ${event.type}")
        }
    }
}
```

- For API details, see [ListExecutions](#) in *AWS SDK for Kotlin API reference*.

## List state machines

The following code example shows how to list Step Functions state machines.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listMachines() {

    SfnClient { region = "us-east-1" }.use { sfnClient ->
        val response = sfnClient.listStateMachines(ListStateMachinesRequest {})
        response.stateMachines?.forEach { machine ->
            println("The name of the state machine is ${machine.name}")
            println("The ARN value is ${machine.stateMachineArn}")
        }
    }
}
```

- For API details, see [ListStateMachines](#) in *AWS SDK for Kotlin API reference*.

## Start a state machine run

The following code example shows how to start a Step Functions state machine run.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun startWorkflow(stateMachineArnVal: String?, jsonFile: String): String? {
    val json = getJSONString(jsonFile)
```

```
// Specify the name of the execution by using a GUID value.  
val uuid = UUID.randomUUID()  
val uuidValue = uuid.toString()  
val request = StartExecutionRequest {  
    input = json  
    stateMachineArn = stateMachineArnVal  
    name = uuidValue  
}  
  
SfnClient { region = "us-east-1" }.use { sfnClient ->  
    val response = sfnClient.startExecution(request)  
    return response.executionArn  
}  
}  
  
private fun getJSONString(path: String): String {  
  
    try {  
        val parser = JSONParser()  
        val data = parser.parse(FileReader(path)) as JSONObject // path to the JSON  
file.  
        return data.toJSONString()  
    } catch (e: IOException) {  
        print(e.message)  
    } catch (e: ParseException) {  
        print(e.message)  
    }  
    return ""  
}
```

- For API details, see [StartExecution](#) in *AWS SDK for Kotlin API reference*.

## AWS Support examples using SDK for Kotlin

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Kotlin with AWS Support.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Get started

#### Hello AWS Support

The following code examples show how to get started using AWS Support.

.NET

#### AWS SDK for .NET

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
using Amazon AWSSupport;  
using Microsoft.Extensions.DependencyInjection;
```

```
using Microsoft.Extensions.Hosting;

public static class HelloSupport
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
        // the AWS Support service.
        // Use your AWS profile name, or leave it blank to use the default profile.
        // You must have one of the following AWS Support plans: Business,
        // Enterprise On-Ramp, or Enterprise. Otherwise, an exception will be thrown.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices(_ =>
                services.AddAWSService<IAmazonAWSSupport>()
            ).Build();

        // Now the client is available for injection.
        var supportClient = host.Services.GetRequiredService<IAmazonAWSSupport>();

        // You can use await and any of the async methods to get a response.
        var response = await supportClient.DescribeServicesAsync();
        Console.WriteLine($"\\tHello AWS Support! There are
{response.Services.Count} services available.");
    }
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for .NET API Reference*.

## Java

### SDK for Java 2.x

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
 * Before running this Java (v2) code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you must have the AWS Business Support Plan to use the AWS Support
 * Java API. For more information, see:
 *
 * https://aws.amazon.com/premiumsupport/plans/
 *
 * This Java example performs the following task:
 *
 * 1. Gets and displays available services.
 *
 *
 * NOTE: To see multiple operations, see SupportScenario.
 */

public class HelloSupport {

    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
```

```
.region(region)
.build();

System.out.println("***** Step 1. Get and display available services.");
displayServices(supportClient);
}

// Return a List that contains a Service name and Category name.
public static void displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
            .language("en")
            .build();

        DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
        List<Service> services = response.services();

        System.out.println("Get the first 10 services");
        int index = 1;
        for (Service service: services) {
            if (index== 11)
                break;

            System.out.println("The Service name is: "+service.name());

            // Display the Categories for this service.
            List<Category> categories = service.categories();
            for (Category cat: categories) {
                System.out.println("The category name is: "+cat.name());
            }
            index++ ;
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for Java 2.x API Reference*.

## Kotlin

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.
```

```
For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
```

In addition, you must have the AWS Business Support Plan to use the AWS Support Java API. For more information, see:

<https://aws.amazon.com/premiumsupport/plans/>

This Kotlin example performs the following task:

```
1. Gets and displays available services.  
*/  
  
suspend fun main() {  
    displaySomeServices()  
}  
  
// Return a List that contains a Service name and Category name.  
suspend fun displaySomeServices() {  
    val servicesRequest = DescribeServicesRequest {  
        language = "en"  
    }  
  
    SupportClient { region = "us-west-2" }.use { supportClient ->  
        val response = supportClient.describeServices(servicesRequest)  
        println("Get the first 10 services")  
        var index = 1  
  
        response.services?.forEach { service ->  
            if (index == 11) {  
                return@forEach  
            }  
  
            println("The Service name is: " + service.name)  
  
            // Get the categories for this service.  
            service.categories?.forEach { cat ->  
                println("The category name is ${cat.name}")  
                index++  
            }  
        }  
    }  
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for Kotlin API reference*.

## Topics

- [Actions \(p. 3652\)](#)
- [Scenarios \(p. 3658\)](#)

## Actions

### Add a communication to a case

The following code example shows how to add an AWS Support communication with an attachment to a support case.

#### SDK for Kotlin

##### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun addAttachSupportCase(caseIdVal: String?, attachmentSetIdVal: String?) {
    val caseRequest = AddCommunicationToCaseRequest {
        caseId = caseIdVal
        attachmentSetId = attachmentSetIdVal
        communicationBody = "Please refer to attachment for details."
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS Support
case")
        } else {
            println("There was an error adding the communication to an AWS Support
case")
        }
    }
}
```

- For API details, see [AddCommunicationToCase](#) in *AWS SDK for Kotlin API reference*.

## Add an attachment to a set

The following code example shows how to add an AWS Support attachment to an attachment set.

### SDK for Kotlin

**Note**

This is prerelease documentation for a feature in preview release. It is subject to change.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal = Attachment {
        fileName = myFile.name
        data = sourceBytes
    }

    val setRequest = AddAttachmentsToSetRequest {
        attachments = listOf(attachmentVal)
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}
```

- For API details, see [AddAttachmentsToSet](#) in *AWS SDK for Kotlin API reference*.

## Create a case

The following code example shows how to create a new AWS Support case.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun createSupportCase(sevCatListVal: List<String>, sevLevelVal: String):  
String? {  
    val serCode = sevCatListVal[0]  
    val caseCategory = sevCatListVal[1]  
    val caseRequest = CreateCaseRequest {  
        categoryCode = caseCategory.lowercase(Locale.getDefault())  
        serviceCode = serCode.lowercase(Locale.getDefault())  
        severityCode = sevLevelVal.lowercase(Locale.getDefault())  
        communicationBody = "Test issue with ${serCode.lowercase(Locale.getDefault())}"  
        subject = "Test case, please ignore"  
        language = "en"  
        issueType = "technical"  
    }  
  
    SupportClient { region = "us-west-2" }.use { supportClient ->  
        val response = supportClient.createCase(caseRequest)  
        return response.caseId  
    }  
}
```

- For API details, see [CreateCase](#) in *AWS SDK for Kotlin API reference*.

## Describe an attachment

The following code example shows how to describe an attachment for an AWS Support case.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun describeAttachment(attachId: String?) {  
    val attachmentRequest = DescribeAttachmentRequest {  
        attachmentId = attachId  
    }  
  
    SupportClient { region = "us-west-2" }.use { supportClient ->  
        val response = supportClient.describeAttachment(attachmentRequest)  
        println("The name of the file is ${response.attachment?.fileName}")  
    }  
}
```

- For API details, see [DescribeAttachment](#) in *AWS SDK for Kotlin API reference*.

## Describe cases

The following code example shows how to describe AWS Support cases.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest = DescribeCasesRequest {
        maxResults = 20
        afterTime = yesterday.toString()
        beforeTime = now.toString()
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}
```

- For API details, see [DescribeCases](#) in *AWS SDK for Kotlin API reference*.

## Describe communications

The following code example shows how to describe AWS Support communications for a case.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest = DescribeCommunicationsRequest {
        caseId = caseIdVal
        maxResults = 10
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->

```

```
        return detail.attachmentId
    }
}
return ""
}
```

- For API details, see [DescribeCommunications](#) in *AWS SDK for Kotlin API reference*.

## Describe services

The following code example shows how to describe the list of AWS services.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest = DescribeServicesRequest {
        language = "en"
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- For API details, see [DescribeServices](#) in *AWS SDK for Kotlin API reference*.

## Describe severity levels

The following code example shows how to describe AWS Support severity levels.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest = DescribeSeverityLevelsRequest {
        language = "en"
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeSeverityLevels(severityLevelsRequest)
        response.severityLevels?.forEach { sevLevel ->
            println("The severity level name is: ${sevLevel.name}")
            if (sevLevel.name == "High") {
                levelName = sevLevel.name!!
            }
        }
    }
    return levelName
}
```

- For API details, see [DescribeSeverityLevels](#) in *AWS SDK for Kotlin API reference*.

## Resolve case

The following code example shows how to resolve an AWS Support case.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest = ResolveCaseRequest {
        caseId = caseIdVal
    }
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}
```

- For API details, see [ResolveCase](#) in *AWS SDK for Kotlin API reference*.

## Scenarios

### Get started with cases

The following code example shows how to:

- Get and display available services and severity levels for cases.
- Create a support case using a selected service, category, and severity level.
- Get and display a list of open cases for the current day.
- Add an attachment set and a communication to the new case.
- Describe the new attachment and communication for the case.
- Resolve the case.
- Get and display a list of resolved cases for the current day.

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/**  
Before running this Kotlin code example, set up your development environment,  
including your credentials.  
  
For more information, see the following documentation topic:  
  
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html  
In addition, you must have the AWS Business Support Plan to use the AWS Support Java  
API. For more information, see:  
  
https://aws.amazon.com/premiumsupport/plans/  
  
This Kotlin example performs the following tasks:  
1. Gets and displays available services.  
2. Gets and displays severity levels.  
3. Creates a support case by using the selected service, category, and severity level.  
4. Gets a list of open cases for the current day.  
5. Creates an attachment set with a generated file.  
6. Adds a communication with the attachment to the support case.  
7. Lists the communications of the support case.  
8. Describes the attachment set included with the communication.  
9. Resolves the support case.  
10. Gets a list of resolved cases for the current day.  
*/  
  
suspend fun main(args: Array<String>) {  
    val usage = """  
    Usage:  
        <fileAttachment>  
    Where:  
        fileAttachment - The file can be a simple saved .txt file to use as an email  
attachment.  
    """
```

```
"""
if (args.size != 1) {
    println(usage)
    exitProcess(0)
}

val fileAttachment = args[0]
println("***** Welcome to the AWS Support case example scenario.")
println("***** Step 1. Get and display available services.")
val sevCatList = displayServices()

println("***** Step 2. Get and display Support severity levels.")
val sevLevel = displaySevLevels()

println("***** Step 3. Create a support case using the selected service, category,
and severity level.")
val caseIdVal = createSupportCase(sevCatList, sevLevel)
if (caseIdVal != null) {
    println("Support case $caseIdVal was successfully created!")
} else {
    println("A support case was not successfully created!")
    exitProcess(1)
}

println("***** Step 4. Get open support cases.")
getOpenCase()

println("***** Step 5. Create an attachment set with a generated file to add to the
case.")
val attachmentSetId = addAttachment(fileAttachment)
println("The Attachment Set id value is $attachmentSetId")

println("***** Step 6. Add communication with the attachment to the support case.")
addAttachSupportCase(caseIdVal, attachmentSetId)

println("***** Step 7. List the communications of the support case.")
val attachId = listCommunications(caseIdVal)
println("The Attachment id value is $attachId")

println("***** Step 8. Describe the attachment set included with the
communication.")
describeAttachment(attachId)

println("***** Step 9. Resolve the support case.")
resolveSupportCase(caseIdVal)

println("***** Step 10. Get a list of resolved cases for the current day.")
getResolvedCase()
println("***** This Scenario has successfully completed")
}

suspend fun getResolvedCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest = DescribeCasesRequest {
        maxResults = 30
        afterTime = yesterday.toString()
        beforeTime = now.toString()
        includeResolvedCases = true
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
    }
}
```

```
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun resolveSupportCase(caseIdVal: String) {
    val caseRequest = ResolveCaseRequest {
        caseId = caseIdVal
    }
    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.resolveCase(caseRequest)
        println("The status of case $caseIdVal is ${response.finalCaseStatus}")
    }
}

suspend fun describeAttachment(attachId: String?) {
    val attachmentRequest = DescribeAttachmentRequest {
        attachmentId = attachId
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeAttachment(attachmentRequest)
        println("The name of the file is ${response.attachment?.fileName}")
    }
}

suspend fun listCommunications(caseIdVal: String?): String? {
    val communicationsRequest = DescribeCommunicationsRequest {
        caseId = caseIdVal
        maxResults = 10
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCommunications(communicationsRequest)
        response.communications?.forEach { comm ->
            println("the body is: " + comm.body)
            comm.attachmentSet?.forEach { detail ->
                return detail.attachmentId
            }
        }
    }
    return ""
}

suspend fun addAttachSupportCase(caseIdVal: String?, attachmentSetIdVal: String?) {
    val caseRequest = AddCommunicationToCaseRequest {
        caseId = caseIdVal
        attachmentSetId = attachmentSetIdVal
        communicationBody = "Please refer to attachment for details."
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addCommunicationToCase(caseRequest)
        if (response.result) {
            println("You have successfully added a communication to an AWS Support
case")
        } else {
            println("There was an error adding the communication to an AWS Support
case")
        }
    }
}
```

```
suspend fun addAttachment(fileAttachment: String): String? {
    val myFile = File(fileAttachment)
    val sourceBytes = (File(fileAttachment).readBytes())
    val attachmentVal = Attachment {
        fileName = myFile.name
        data = sourceBytes
    }

    val setRequest = AddAttachmentsToSetRequest {
        attachments = listOf(attachmentVal)
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.addAttachmentsToSet(setRequest)
        return response.attachmentSetId
    }
}

suspend fun getOpenCase() {
    // Specify the start and end time.
    val now = Instant.now()
    LocalDate.now()
    val yesterday = now.minus(1, ChronoUnit.DAYS)
    val describeCasesRequest = DescribeCasesRequest {
        maxResults = 20
        afterTime = yesterday.toString()
        beforeTime = now.toString()
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeCases(describeCasesRequest)
        response.cases?.forEach { sinCase ->
            println("The case status is ${sinCase.status}")
            println("The case Id is ${sinCase.caseId}")
            println("The case subject is ${sinCase.subject}")
        }
    }
}

suspend fun createSupportCase(sevCatListVal: List<String>, sevLevelVal: String): String? {
    val serCode = sevCatListVal[0]
    val caseCategory = sevCatListVal[1]
    val caseRequest = CreateCaseRequest {
        categoryCode = caseCategory.lowercase(Locale.getDefault())
        serviceCode = serCode.lowercase(Locale.getDefault())
        severityCode = sevLevelVal.lowercase(Locale.getDefault())
        communicationBody = "Test issue with ${serCode.lowercase(Locale.getDefault())}"
        subject = "Test case, please ignore"
        language = "en"
        issueType = "technical"
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.createCase(caseRequest)
        return response.caseId
    }
}

suspend fun displaySevLevels(): String {
    var levelName = ""
    val severityLevelsRequest = DescribeSeverityLevelsRequest {
        language = "en"
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
```

```
    val response = supportClient.describeSeverityLevels(severityLevelsRequest)
    response.severityLevels?.forEach { sevLevel ->
        println("The severity level name is: ${sevLevel.name}")
        if (sevLevel.name == "High") {
            levelName = sevLevel.name!!
        }
    }
    return levelName
}

// Return a List that contains a Service name and Category name.
suspend fun displayServices(): List<String> {
    var serviceCode = ""
    var catName = ""
    val sevCatList = mutableListOf<String>()
    val servicesRequest = DescribeServicesRequest {
        language = "en"
    }

    SupportClient { region = "us-west-2" }.use { supportClient ->
        val response = supportClient.describeServices(servicesRequest)
        println("Get the first 10 services")
        var index = 1

        response.services?.forEach { service ->
            if (index == 11) {
                return@forEach
            }

            println("The Service name is ${service.name}")
            if (service.name == "Account") {
                serviceCode = service.code.toString()
            }

            // Get the categories for this service.
            service.categories?.forEach { cat ->
                println("The category name is ${cat.name}")
                if (cat.name == "Security") {
                    catName = cat.name!!
                }
            }
            index++
        }
    }

    // Push the two values to the list.
    serviceCode.let { sevCatList.add(it) }
    catName.let { sevCatList.add(it) }
    return sevCatList
}
```

- For API details, see the following topics in *AWS SDK for Kotlin API reference*.

- [AddAttachmentsToSet](#)
- [AddCommunicationToCase](#)
- [CreateCase](#)
- [DescribeAttachment](#)
- [DescribeCases](#)
- [DescribeCommunications](#)
- [DescribeServices](#)
- [DescribeSeverityLevels](#)

- [ResolveCase](#)

## Cross-service examples using SDK for Kotlin

The following sample applications use the AWS SDK for Kotlin to work across multiple AWS services.

### Examples

- [Build an application to submit data to a DynamoDB table \(p. 3663\)](#)
- [Build a publish and subscription application that translates messages \(p. 3663\)](#)
- [Create a web application that sends and retrieves messages by using the Amazon Simple Queue Service API \(p. 3664\)](#)
- [Create a web application to track DynamoDB data \(p. 3664\)](#)
- [Create an Amazon Redshift item tracker \(p. 3664\)](#)
- [Create an Aurora Serverless work item tracker \(p. 3664\)](#)
- [Detect objects in images with Amazon Rekognition using an AWS SDK \(p. 3665\)](#)

## Build an application to submit data to a DynamoDB table

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to create a native Android application that submits data using the Amazon DynamoDB Kotlin API and sends a text message using the Amazon SNS Kotlin API.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon SNS

## Build a publish and subscription application that translates messages

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to use the Amazon SNS Kotlin API to create an application that has subscription and publish functionality. In addition, this example application also translates messages.

For complete source code and instructions on how to create a web app, see the full example on [GitHub](#).

For complete source code and instructions on how to create a native Android app, see the full example on [GitHub](#).

### Services used in this example

- Amazon SNS
- Amazon Translate

## Create a web application that sends and retrieves messages by using the Amazon Simple Queue Service API

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to use the Amazon Simple Queue Service API to develop a Spring REST API that sends and retrieves messages.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Comprehend
- Amazon SQS

## Create a web application to track DynamoDB data

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to use the Amazon DynamoDB API to create a dynamic web application that tracks DynamoDB work data.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- DynamoDB
- Amazon SES

## Create an Amazon Redshift item tracker

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to create a web application that tracks and reports on work items stored in an Amazon Redshift database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Redshift data and for use by a React application, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Redshift
- Amazon SES

## Create an Aurora Serverless work item tracker

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to create a web application that tracks and reports on work items stored in an Amazon RDS database.

For complete source code and instructions on how to set up a Spring REST API that queries Amazon Aurora Serverless data and for use by a React application, see the full example on [GitHub](#).

#### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

## Detect objects in images with Amazon Rekognition using an AWS SDK

### SDK for Kotlin

#### Note

This is prerelease documentation for a feature in preview release. It is subject to change.

Shows how to use Amazon Rekognition Kotlin API to create an app that uses Amazon Rekognition to identify objects by category in images located in an Amazon Simple Storage Service (Amazon S3) bucket. The app sends the admin an email notification with the results using Amazon Simple Email Service (Amazon SES).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Code examples for SDK for PHP

The code examples in this topic show you how to use the AWS SDK for PHP with AWS.

### Examples

- [Single-service actions and scenarios using SDK for PHP \(p. 3665\)](#)
- [Cross-service examples using SDK for PHP \(p. 3735\)](#)

## Single-service actions and scenarios using SDK for PHP

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for PHP with AWS services.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Services

- [API Gateway examples using SDK for PHP \(p. 3666\)](#)
- [DynamoDB examples using SDK for PHP \(p. 3669\)](#)
- [AWS Glue examples using SDK for PHP \(p. 3687\)](#)
- [IAM examples using SDK for PHP \(p. 3700\)](#)
- [Lambda examples using SDK for PHP \(p. 3711\)](#)
- [Amazon S3 examples using SDK for PHP \(p. 3717\)](#)
- [Amazon SNS examples using SDK for PHP \(p. 3724\)](#)

## API Gateway examples using SDK for PHP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for PHP with Amazon API Gateway.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3666\)](#)

## Actions

### Get base path mapping

The following code example shows how to get an API Gateway base path mapping.

#### SDK for PHP

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\ApiGateway\ApiGatewayClient;
use Aws\Exception\AwsException;

/*
 * Purpose: Gets the base path mapping for a custom domain name in
 * Amazon API Gateway.
 *
 * Prerequisites: A custom domain name in API Gateway. For more information,
 * see "Custom Domain Names" in the Amazon API Gateway Developer Guide.
 *
 * Inputs:
 * - $apiGatewayClient: An initialized AWS SDK for PHP API client for
 *   API Gateway.
 * - $basePath: The base path name that callers must provide as part of the
 *   URL after the domain name.
 * - $domainName: The custom domain name for the base path mapping.
 *
 * Returns: The base path mapping, if available; otherwise, the error message.
 */
function getBasePathMapping($apiGatewayClient, $basePath, $domainName)
```

```
{  
    try {  
        $result = $apiGatewayClient->getBasePathMapping([  
            'basePath' => $basePath,  
            'domainName' => $domainName,  
        ]);  
        return 'The base path mapping\'s effective URI is: ' .  
        $result['@metadata']['effectiveUri'];  
    } catch (AwsException $e) {  
        return 'Error: ' . $e['message'];  
    }  
}  
  
function getsTheBasePathMapping()  
{  
    $apiGatewayClient = new ApiGatewayClient([  
        'profile' => 'default',  
        'region' => 'us-east-1',  
        'version' => '2015-07-09'  
    ]);  
  
    echo getBasePathMapping($apiGatewayClient, '(none)', 'example.com');  
}  
  
// Uncomment the following line to run this code in an AWS account.  
// getsTheBasePathMapping();
```

- For API details, see [GetBasePathMapping](#) in *AWS SDK for PHP API Reference*.

## List base path mapping

The following code example shows how to list an API Gateway base path mapping.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';  
  
use Aws\ApiGateway\ApiGatewayClient;  
use Aws\Exception\AwsException;  
  
/* ///////////////////////////////// * Purpose: Lists the base path mapping for a custom domain name in  
* Amazon API Gateway.  
*  
* Prerequisites: A custom domain name in API Gateway. For more information,  
* see "Custom Domain Names" in the Amazon API Gateway Developer Guide.  
*  
* Inputs:  
* - $apiGatewayClient: An initialized AWS SDK for PHP API client for  
*   API Gateway.  
* - $domainName: The custom domain name for the base path mappings.  
*  
* Returns: Information about the base path mappings, if available;  
* otherwise, the error message.  
* ////////////////////////////// */  
  
function listBasePathMappings($apiGatewayClient, $domainName)
```

```
{  
    try {  
        $result = $apiGatewayClient->getBasePathMappings([  
            'domainName' => $domainName  
        ]);  
        return 'The base path mapping(s) effective URI is: ' .  
            $result['@metadata']['effectiveUri'];  
    } catch (AwsException $e) {  
        return 'Error: ' . $e['message'];  
    }  
}  
  
function listThebasePathMappings()  
{  
    $apiGatewayClient = new ApiGatewayClient([  
        'profile' => 'default',  
        'region' => 'us-east-1',  
        'version' => '2015-07-09'  
    ]);  
  
    echo listBasePathMappings($apiGatewayClient, 'example.com');  
}  
  
// Uncomment the following line to run this code in an AWS account.  
// listThebasePathMappings();
```

- For API details, see [ListBasePathMappings](#) in *AWS SDK for PHP API Reference*.

## Update base path mapping

The following code example shows how to update an API Gateway base path mapping.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';  
  
use Aws\ApiGateway\ApiGatewayClient;  
use Aws\Exception\AwsException;  
  
/* ////////////////////////////// */  
*  
* Purpose: Updates the base path mapping for a custom domain name  
* in Amazon API Gateway.  
*  
* Inputs:  
* - $apiGatewayClient: An initialized AWS SDK for PHP API client for  
*   API Gateway.  
* - $basePath: The base path name that callers must provide as part of the  
*   URL after the domain name.  
* - $domainName: The custom domain name for the base path mapping.  
* - $patchOperations: The base path update operations to apply.  
*  
* Returns: Information about the updated base path mapping, if available;  
* otherwise, the error message.  
* ////////////////////////////// */  
  
function updatebasePathMapping($apiGatewayClient, $basePath, $domainName,  
    $patchOperations)
```

```
{  
    try {  
        $result = $apiGatewayClient->updateBasePathMapping([  
            'basePath' => $basePath,  
            'domainName' => $domainName,  
            'patchOperations' => $patchOperations  
        ]);  
        return 'The updated base path\'s URI is: ' .  
            $result['@metadata']['effectiveUri'];  
    } catch (AwsException $e) {  
        return 'Error: ' . $e['message'];  
    }  
}  
  
function updateTheBasePathMapping()  
{  
    $patchOperations = array([  
        'op' => 'replace',  
        'path' => '/stage',  
        'value' => 'stage2'  
    ]);  
  
    $apiGatewayClient = new ApiGatewayClient([  
        'profile' => 'default',  
        'region' => 'us-east-1',  
        'version' => '2015-07-09'  
    ]);  
  
    echo updateBasePathMapping(  
        $apiGatewayClient,  
        '(none)',  
        'example.com',  
        $patchOperations);  
}  
  
// Uncomment the following line to run this code in an AWS account.  
// updateTheBasePathMapping();
```

- For API details, see [UpdateBasePathMapping](#) in *AWS SDK for PHP API Reference*.

## DynamoDB examples using SDK for PHP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for PHP with Amazon DynamoDB.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3669\)](#)
- [Scenarios \(p. 3677\)](#)

## Actions

### Create a table

The following code example shows how to create a DynamoDB table.

## SDK for PHP

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a table.

```
$tableName = "ddb_demo_table_$uuid";
$service->createTable(
    $tableName,
    [
        new DynamoDBAttribute('year', 'N', 'HASH'),
        new DynamoDBAttribute('title', 'S', 'RANGE')
    ],
    [
        new DynamoDBAttribute('year', 'N', 'HASH'),
        new DynamoDBAttribute('title', 'S', 'RANGE')
    ]
);

public function createTable(string $tableName, array $attributes)
{
    $keySchema = [];
    $attributeDefinitions = [];
    foreach ($attributes as $attribute) {
        if (is_a($attribute, DynamoDBAttribute::class)) {
            $keySchema[] = ['AttributeName' => $attribute->AttributeName, 'KeyType' => $attribute->KeyType];
            $attributeDefinitions[] = ['AttributeName' => $attribute->AttributeName, 'AttributeType' => $attribute->AttributeType];
        }
    }

    $this->dynamoDbClient->createTable([
        'TableName' => $tableName,
        'KeySchema' => $keySchema,
        'AttributeDefinitions' => $attributeDefinitions,
        'ProvisionedThroughput' => ['ReadCapacityUnits' => 10, 'WriteCapacityUnits' => 10],
    ]);
}
```

- For API details, see [CreateTable](#) in *AWS SDK for PHP API Reference*.

## Delete a table

The following code example shows how to delete a DynamoDB table.

## SDK for PHP

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function deleteTable(string $TableName)
{
    $this->customWaiter(function () use ($TableName) {
        return $this->dynamoDbClient->deleteTable([
            'TableName' => $TableName,
        ]);
    });
}
```

- For API details, see [DeleteTable](#) in *AWS SDK for PHP API Reference*.

## Delete an item from a table

The following code example shows how to delete an item from a DynamoDB table.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$key = [
    'Item' => [
        'title' => [
            'S' => $movieName,
        ],
        'year' => [
            'N' => $movieYear,
        ],
    ],
];
];

(service->deleteItemByKey($tableName, $key);
echo "But, bad news, this was a trap. That movie has now been deleted because
of your rating...harsh.\n";

public function deleteItemByKey(string $tableName, array $key)
{
    $this->dynamoDbClient->deleteItem([
        'Key' => $key['Item'],
        'TableName' => $tableName,
    ]);
}
```

- For API details, see [DeleteItem](#) in *AWS SDK for PHP API Reference*.

## Get an item from a table

The following code example shows how to get an item from a DynamoDB table.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$movie = $service->getItemByKey($tableName, $key);
echo "\nThe movie {$movie['Item']['title']['S']} was released in
{$movie['Item']['year']['N']}.\n";

public function getItemByKey(string $tableName, array $key)
{
    return $this->dynamoDbClient->getItem([
```

```
        'Key' => $key['Item'],
        'TableName' => $tableName,
    ]);
}
```

- For API details, see [GetItem](#) in *AWS SDK for PHP API Reference*.

## List tables

The following code example shows how to list DynamoDB tables.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function listTables($exclusiveStartTableName = "", $limit = 100)
{
    $this->dynamoDbClient->listTables([
        'ExclusiveStartTableName' => $exclusiveStartTableName,
        'Limit' => $limit,
    ]);
}
```

- For API details, see [ListTables](#) in *AWS SDK for PHP API Reference*.

## Put an item in a table

The following code example shows how to put an item in a DynamoDB table.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
echo "What's the name of the last movie you watched?\n";
while (empty($movieName)) {
    $movieName = testable_readline("Movie name: ");
}
echo "And what year was it released?\n";
$movieYear = "year";
while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
    $movieYear = testable_readline("Year released: ");
}

$service->putItem([
    'Item' => [
        'year' => [
            'N' => "$movieYear",
        ],
        'title' => [
            'S' => $movieName,
        ],
    ],
    'TableName' => $tableName,
])
```

```
    ]);

    public function putItem(array $array)
    {
        $this->dynamoDbClient->putItem($array);
    }
```

- For API details, see [PutItem](#) in *AWS SDK for PHP API Reference*.

## Query a table

The following code example shows how to query a DynamoDB table.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$birthKey = [
    'Key' => [
        'year' => [
            'N' => "$birthYear",
        ],
    ],
];
$result = $service->query($tableName, $birthKey);

public function query(string $tableName, $key)
{
    $expressionAttributeValues = [];
    $expressionAttributeNames = [];
    $keyConditionExpression = "";
    $index = 1;
    foreach ($key as $name => $value) {
        $keyConditionExpression .= "#" . array_key_first($value) . " = :v$index,";
        $expressionAttributeNames["#" . array_key_first($value)] =
array_key_first($value);
        $hold = array_pop($value);
        $expressionAttributeValues[":v$index"] = [
            array_key_first($hold) => array_pop($hold),
        ];
    }
    $keyConditionExpression = substr($keyConditionExpression, 0, -1);
    $query = [
        'ExpressionAttributeValues' => $expressionAttributeValues,
        'ExpressionAttributeNames' => $expressionAttributeNames,
        'KeyConditionExpression' => $keyConditionExpression,
        'TableName' => $tableName,
    ];
    return $this->dynamoDbClient->query($query);
}
```

- For API details, see [Query](#) in *AWS SDK for PHP API Reference*.

## Run a PartiQL statement

The following code example shows how to run a PartiQL statement on a DynamoDB table.

## SDK for PHP

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function insertItemByPartiQL(string $statement, array $parameters)
{
    $this->dynamoDbClient->executeStatement([
        'Statement' => "$statement",
        'Parameters' => $parameters,
    ]);
}

public function getItemByPartiQL(string $tableName, array $key): Result
{
    list($statement, $parameters) = $this->buildStatementAndParameters("SELECT",
$tableName, $key['Item']);

    return $this->dynamoDbClient->executeStatement([
        'Parameters' => $parameters,
        'Statement' => $statement,
    ]);
}

public function updateItemByPartiQL(string $statement, array $parameters)
{
    $this->dynamoDbClient->executeStatement([
        'Statement' => $statement,
        'Parameters' => $parameters,
    ]);
}

public function deleteItemByPartiQL(string $statement, array $parameters)
{
    $this->dynamoDbClient->executeStatement([
        'Statement' => $statement,
        'Parameters' => $parameters,
    ]);
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for PHP API Reference*.

## Run batches of PartiQL statements

The following code example shows how to run batches of PartiQL statements on a DynamoDB table.

## SDK for PHP

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function getItemByPartiQLBatch(string $tableName, array $keys): Result
{
    $statements = [];
    foreach ($keys as $key) {
        list($statement, $parameters) = $this-
>buildStatementAndParameters("SELECT", $tableName, $key['Item']);
        $statements[] = [
            'Statement' => "$statement",

```

```
        'Parameters' => $parameters,
    ];
}

return $this->dynamoDbClient->batchExecuteStatement([
    'Statements' => $statements,
]);
}

public function insertItemByPartiQLBatch(string $statement, array $parameters)
{
    $this->dynamoDbClient->batchExecuteStatement([
        'Statements' => [
            [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ],
        ],
    ]);
}

public function updateItemByPartiQLBatch(string $statement, array $parameters)
{
    $this->dynamoDbClient->batchExecuteStatement([
        'Statements' => [
            [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ],
        ],
    ]);
}

public function deleteItemByPartiQLBatch(string $statement, array $parameters)
{
    $this->dynamoDbClient->batchExecuteStatement([
        'Statements' => [
            [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ],
        ],
    ]);
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for PHP API Reference*.

## Scan a table

The following code example shows how to scan a DynamoDB table.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$yearsKey = [
    'Key' => [
        'year' => [
            'N' => [
                'minRange' => 1990,
```

```
        'maxRange' => 1999,
    ],
],
];
$filter = "year between 1990 and 1999";
echo "\nHere's a list of all the movies released in the 90s:\n";
$result = $service->scan($tableName, $yearsKey, $filter);
foreach ($result['Items'] as $movie) {
    $movie = $marshal->unmarshalItem($movie);
    echo $movie['title'] . "\n";
}

public function scan(string $tableName, array $key, string $filters)
{
    $query = [
        'ExpressionAttributeNames' => ['#year' => 'year'],
        'ExpressionAttributeValues' => [
            ':min' => ['N' => '1990'],
            ':max' => ['N' => '1999'],
        ],
        'FilterExpression' => "#year between :min and :max",
        'TableName' => $tableName,
    ];
    return $this->dynamoDbClient->scan($query);
}
```

- For API details, see [Scan](#) in *AWS SDK for PHP API Reference*.

## Update an item in a table

The following code example shows how to update an item in a DynamoDB table.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
$rating = 0;
while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 || $rating > 10) {
    $rating = testable_readline("Rating (1-10): ");
}
	service->updateItemAttributeByKey($tableName, $key, 'rating', 'N', $rating);

public function updateItemAttributeByKey(
    string $tableName,
    array $key,
    string $attributeName,
    string $attributeType,
    string $newValue
) {
    $this->dynamoDbClient->updateItem([
        'Key' => $key['Item'],
        'TableName' => $tableName,
        'UpdateExpression' => "set #NV=:NV",
        'ExpressionAttributeNames' => [
            '#NV' => $attributeName,
        ],
        'ExpressionAttributeValues' => [
            ':NV' => $newValue,
        ],
    ]);
}
```

```
        ':NV' => [
            $attributeType => $newValue
        ],
    ],
});
```

- For API details, see [UpdateItem](#) in *AWS SDK for PHP API Reference*.

## Write a batch of items

The following code example shows how to write a batch of DynamoDB items.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function writeBatch(string $TableName, array $Batch, int $depth = 2)
{
    if (--$depth <= 0) {
        throw new Exception("Max depth exceeded. Please try with fewer batch items or increase depth.");
    }

    $marshal = new Marshaler();
    $total = 0;
    foreach (array_chunk($Batch, 25) as $Items) {
        foreach ($Items as $Item) {
            $BatchWrite['RequestItems'][$TableName][] = ['PutRequest' => ['Item' => $marshal->marshalItem($Item)]];
        }
        try {
            echo "Batching another " . count($Items) . " for a total of " . ($total += count($Items)) . " items!\n";
            $response = $this->dynamoDbClient->batchWriteItem($BatchWrite);
            $BatchWrite = [];
        } catch (Exception $e) {
            echo "uh oh...";
            echo $e->getMessage();
            die();
        }
        if ($total >= 250) {
            echo "250 movies is probably enough. Right? We can stop there.\n";
            break;
        }
    }
}
```

- For API details, see [BatchWriteItem](#) in *AWS SDK for PHP API Reference*.

## Scenarios

### Get started using tables, items, and queries

The following code example shows how to:

- Create a table that can hold movie data.

- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.
- Delete a movie from the table.
- Delete the table.

## SDK for PHP

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace DynamoDb\Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;
use DynamoDb\DynamoDBService;

class GettingStartedWithDynamoDB
{
    public function run()
    {
        echo "-----\n";
        print("Welcome to the Amazon DynamoDB getting started demo using PHP!\n");
        echo "-----\n";

        $uuid = uniqid();
        $service = new DynamoDBService();

        $tableName = "ddb_demo_table_{$uuid}";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' => $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }

        $service->putItem([
            'Item' => [
                'year' => [
                    'N' => "{$movieYear}",
                ],
                'title' => [

```

```

        'S' => $movieName,
    ],
],
'TableName' => $tableName,
]);

echo "How would you rate the movie from 1-10?\n";
$rating = 0;
while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 || $rating > 10) {
    $rating = testable_readline("Rating (1-10): ");
}
echo "What was the movie about?\n";
while (empty($plot)) {
    $plot = testable_readline("Plot summary: ");
}
$key = [
    'Item' => [
        'title' => [
            'S' => $movieName,
        ],
        'year' => [
            'N' => $movieYear,
        ],
    ]
];
$attributes = ["rating" =>
    [
        'AttributeName' => 'rating',
        'AttributeType' => 'N',
        'Value' => $rating,
    ],
    'plot' => [
        'AttributeName' => 'plot',
        'AttributeType' => 'S',
        'Value' => $plot,
    ]
];
$service->updateItemAttributesByKey($tableName, $key, $attributes);
echo "Movie added and updated.";

$batch = json_decode(loadMovieData());

$limit = 0;
$service->writeBatch($tableName, $batch);

$movie = $service->getItemByKey($tableName, $key);
echo "\nThe movie {$movie['Item']['title']['S']} was released in {$movie['Item']['year']['N']}.\n";
echo "What rating would you like to give {$movie['Item']['title']['S']}?\n";
$rating = 0;
while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 || $rating > 10) {
    $rating = testable_readline("Rating (1-10): ");
}
$service->updateItemAttributeByKey($tableName, $key, 'rating', 'N', $rating);

$movie = $service->getItemByKey($tableName, $key);
echo "Ok, you have rated {$movie['Item']['title']['S']} as a {$movie['Item']['rating']['N']}.\n";

$service->deleteItemByKey($tableName, $key);
echo "But, bad news, this was a trap. That movie has now been deleted because of your rating...harsh.\n";

```

```

echo "That's okay though. The book was better. Now, for something lighter, in
what year were you born?\n";
$birthYear = "not a number";
while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
    $birthYear = testable_readline("Birth year: ");
}
$birthKey = [
    'Key' => [
        'year' => [
            'N' => "$birthYear",
        ],
    ],
];
$result = $service->query($tableName, $birthKey);
$marshal = new Marshaler();
echo "Here are the movies in our collection released the year you were born:
\n";
$oops = "Oops! There were no movies released in that year (that we know of).
\n";
$display = "";
foreach ($result['Items'] as $movie) {
    $movie = $marshal->unmarshalItem($movie);
    $display .= $movie['title'] . "\n";
}
echo ($display) ?: $oops;

$yearsKey = [
    'Key' => [
        'year' => [
            'N' => [
                'minRange' => 1990,
                'maxRange' => 1999,
            ],
        ],
    ],
];
$filter = "year between 1990 and 1999";
echo "\nHere's a list of all the movies released in the 90s:\n";
$result = $service->scan($tableName, $yearsKey, $filter);
foreach ($result['Items'] as $movie) {
    $movie = $marshal->unmarshalItem($movie);
    echo $movie['title'] . "\n";
}
echo "\nCleaning up this demo by deleting table $tableName...\n";
$service->deleteTable($tableName);
}

```

- For API details, see the following topics in *AWS SDK for PHP API Reference*.

- [BatchWriter](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## Query a table by using batches of PartiQL statements

The following code example shows how to:

- Get a batch of items by running multiple SELECT statements.
- Add a batch of items by running multiple INSERT statements.
- Update a batch of items by running multiple UPDATE statements.
- Delete a batch of items by running multiple DELETE statements.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

class GettingStartedWithPartiQLBatch
{
    public function run()
    {
        echo("-----\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using PHP!\n");
        echo("-----\n");

        $uuid = uniqid();
        $service = new DynamoDb\DynamoDBService();

        $tableName = "partiql_demo_table_{$uuid}";
        $service->createTable(
            $tableName,
            [
                new DynamoDBAttribute('year', 'N', 'HASH'),
                new DynamoDBAttribute('title', 'S', 'RANGE')
            ]
        );

        echo "Waiting for table...";
        $service->dynamoDbClient->waitUntil("TableExists", ['TableName' => $tableName]);
        echo "table $tableName found!\n";

        echo "What's the name of the last movie you watched?\n";
        while (empty($movieName)) {
            $movieName = testable_readline("Movie name: ");
        }
        echo "And what year was it released?\n";
        $movieYear = "year";
        while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
            $movieYear = testable_readline("Year released: ");
        }
        $key = [
            'Item' => [
                'year' => [
                    'N' => "$movieYear",
                ],
            ],
        ];
    }
}
```

```

        'title' => [
            'S' => $movieName,
        ],
    ],
];
list($statement, $parameters) = $service->buildStatementAndParameters("INSERT",
$tableName, $key);
$service->insertItemByPartiQLBatch($statement, $parameters);

echo "How would you rate the movie from 1-10?\n";
$rating = 0;
while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 || $rating > 10) {
    $rating = testable_readline("Rating (1-10): ");
}
echo "What was the movie about?\n";
while (empty($plot)) {
    $plot = testable_readline("Plot summary: ");
}
$attributes = [
    new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
    new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
];
list($statement, $parameters) = $service->buildStatementAndParameters("UPDATE",
$tableName, $key, $attributes);
$service->updateItemByPartiQLBatch($statement, $parameters);
echo "Movie added and updated.\n";

$batch = json_decode(loadMovieData());

$service->writeBatch($tableName, $batch);

$movie = $service->getItemByPartiQLBatch($tableName, [$key]);
echo "\nThe movie {$movie['Responses'][0]['Item']['title']['S']} was released in {$movie['Responses'][0]['Item']['year']['N']}.\n";
echo "What rating would you like to give {$movie['Responses'][0]['Item']['title']['S']}?\n";
$rating = 0;
while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 || $rating > 10) {
    $rating = testable_readline("Rating (1-10): ");
}
$attributes = [
    new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
    new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
];
list($statement, $parameters) = $service->buildStatementAndParameters("UPDATE",
$tableName, $key, $attributes);
$service->updateItemByPartiQLBatch($statement, $parameters);

$movie = $service->getItemByPartiQLBatch($tableName, [$key]);
echo "Okay, you have rated {$movie['Responses'][0]['Item']['title']['S']} as a {$movie['Responses'][0]['Item']['rating']['N']}.\n";

$service->deleteItemByPartiQLBatch($statement, $parameters);
echo "But, bad news, this was a trap. That movie has now been deleted because of your rating...harsh.\n";

echo "That's okay though. The book was better. Now, for something lighter, in what year were you born?\n";
$birthYear = "not a number";
while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
    $birthYear = testable_readline("Birth year: ");
}
$birthKey = [

```

```

        'Key' => [
            'year' => [
                'N' => "$birthYear",
            ],
        ],
    ],
];
$result = $service->query($tableName, $birthKey);
$marshal = new Marshaler();
echo "Here are the movies in our collection released the year you were born:
\n";
$oops = "Oops! There were no movies released in that year (that we know of).
\n";
$display = "";
foreach ($result['Items'] as $movie) {
    $movie = $marshal->unmarshalItem($movie);
    $display .= $movie['title'] . "\n";
}
echo ($display) ?: $oops;

$yearsKey = [
    'Key' => [
        'year' => [
            'N' => [
                'minRange' => 1990,
                'maxRange' => 1999,
            ],
        ],
    ],
];
$filter = "year between 1990 and 1999";
echo "\nHere's a list of all the movies released in the 90s:\n";
$result = $service->scan($tableName, $yearsKey, $filter);
foreach ($result['Items'] as $movie) {
    $movie = $marshal->unmarshalItem($movie);
    echo $movie['title'] . "\n";
}

echo "\nCleaning up this demo by deleting table $tableName...\n";
 getService->deleteTable($tableName);
}

public function insertItemByPartiQLBatch(string $statement, array $parameters)
{
    $this->dynamoDbClient->batchExecuteStatement([
        'Statements' => [
            [
                'Statement' => "$statement",
                'Parameters' => $parameters,
            ],
        ],
    ]);
}

public function getItemByPartiQLBatch(string $tableName, array $keys): Result
{
    $statements = [];
    foreach ($keys as $key) {
        list($statement, $parameters) = $this->buildStatementAndParameters("SELECT", $tableName, $key['Item']);
        $statements[] = [
            'Statement' => "$statement",
            'Parameters' => $parameters,
        ];
    }
}

```

```
        return $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => $statements,
        ]);
    }

    public function updateItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }

    public function deleteItemByPartiQLBatch(string $statement, array $parameters)
    {
        $this->dynamoDbClient->batchExecuteStatement([
            'Statements' => [
                [
                    'Statement' => "$statement",
                    'Parameters' => $parameters,
                ],
            ],
        ]);
    }
}
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for PHP API Reference*.

## Query a table using PartiQL

The following code example shows how to:

- Get an item by running a SELECT statement.
- Add an item by running an INSERT statement.
- Update an item by running an UPDATE statement.
- Delete an item by running a DELETE statement.

## SDK for PHP

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace DynamoDb\PartiQL_Basics;

use Aws\DynamoDb\Marshaler;
use DynamoDb;
use DynamoDb\DynamoDBAttribute;

class GettingStartedWithPartiQL
{
    public function run()
    {
        echo("-----\n");
        print("Welcome to the Amazon DynamoDB - PartiQL getting started demo using PHP!\n");
    }
}
```

```
echo("-----\n");

$uuid = uniqid();
 getService = new DynamoDb\DynamoDBService();

$tableName = "partiql_demo_table_{$uuid}";
 getService->createTable(
    $tableName,
    [
        new DynamoDBAttribute('year', 'N', 'HASH'),
        new DynamoDBAttribute('title', 'S', 'RANGE')
    ]
);

echo "Waiting for table...";
 getService->dynamoDbClient->waitUntil("TableExists", ['TableName' =>
$tableName]);
echo "table $tableName found!\n";

echo "What's the name of the last movie you watched?\n";
while (empty($movieName)) {
    $movieName = testable_readline("Movie name: ");
}
echo "And what year was it released?\n";
$movieYear = "year";
while (!is_numeric($movieYear) || intval($movieYear) != $movieYear) {
    $movieYear = testable_readline("Year released: ");
}
$key = [
    'Item' => [
        'year' => [
            'N' => "$movieYear",
        ],
        'title' => [
            'S' => $movieName,
        ],
    ],
];
list($statement, $parameters) = $service->buildStatementAndParameters("INSERT",
$tableName, $key);
 getService->insertItemByPartiQL($statement, $parameters);

echo "How would you rate the movie from 1-10?\n";
$rating = 0;
while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 ||
$rating > 10) {
    $rating = testable_readline("Rating (1-10): ");
}
echo "What was the movie about?\n";
while (empty($plot)) {
    $plot = testable_readline("Plot summary: ");
}
$attributes = [
    new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
    new DynamoDBAttribute('plot', 'S', 'RANGE', $plot),
];
list($statement, $parameters) = $service->buildStatementAndParameters("UPDATE",
$tableName, $key, $attributes);
 getService->updateItemByPartiQL($statement, $parameters);
echo "Movie added and updated.\n";

$batch = json_decode(loadMovieData());
```

```

$service->writeBatch($tableName, $batch);

$movie = $service->getItemByPartiQL($tableName, $key);
echo "\nThe movie {$movie['Items'][0]['title']['S']} was released in
{$movie['Items'][0]['year']['N']}.\n";
echo "What rating would you like to give {$movie['Items'][0]['title']['S']}?\n";
$rating = 0;
while (!is_numeric($rating) || intval($rating) != $rating || $rating < 1 || $rating > 10) {
    $rating = testable_readline("Rating (1-10): ");
}
$attributes = [
    new DynamoDBAttribute('rating', 'N', 'HASH', $rating),
    new DynamoDBAttribute('plot', 'S', 'RANGE', $plot)
];
list($statement, $parameters) = $service->buildStatementAndParameters("UPDATE",
$tableName, $key, $attributes);
$service->updateItemByPartiQL($statement, $parameters);

$movie = $service->getItemByPartiQL($tableName, $key);
echo "Okay, you have rated {$movie['Items'][0]['title']['S']} as a
{$movie['Items'][0]['rating']['N']}\n";

$service->deleteItemByPartiQL($statement, $parameters);
echo "But, bad news, this was a trap. That movie has now been deleted because
of your rating...harsh.\n";

echo "That's okay though. The book was better. Now, for something lighter, in
what year were you born?\n";
$birthYear = "not a number";
while (!is_numeric($birthYear) || $birthYear >= date("Y")) {
    $birthYear = testable_readline("Birth year: ");
}
$birthKey = [
    'Key' => [
        'year' => [
            'N' => "$birthYear",
        ],
    ],
];
$result = $service->query($tableName, $birthKey);
$marshal = new Marshaler();
echo "Here are the movies in our collection released the year you were born:
\n";
$oops = "Oops! There were no movies released in that year (that we know of).
\n";
$display = "";
foreach ($result['Items'] as $movie) {
    $movie = $marshal->unmarshalItem($movie);
    $display .= $movie['title'] . "\n";
}
echo ($display) ?: $oops;

$yearsKey = [
    'Key' => [
        'year' => [
            'N' => [
                'minRange' => 1990,
                'maxRange' => 1999,
            ],
        ],
    ],
];
$filter = "year between 1990 and 1999";
echo "\nHere's a list of all the movies released in the 90s:\n";

```

```
$result = $service->scan($tableName, $yearsKey, $filter);
foreach ($result['Items'] as $movie) {
    $movie = $marshal->unmarshalItem($movie);
    echo $movie['title'] . "\n";
}

echo "\nCleaning up this demo by deleting table $tableName...\n";
$service->deleteTable($tableName);
}

public function insertItemByPartiQL(string $statement, array $parameters)
{
    $this->dynamoDbClient->executeStatement([
        'Statement' => "$statement",
        'Parameters' => $parameters,
    ]);
}

public function getItemByPartiQL(string $tableName, array $key): Result
{
    list($statement, $parameters) = $this->buildStatementAndParameters("SELECT",
$tableName, $key['Item']);

    return $this->dynamoDbClient->executeStatement([
        'Parameters' => $parameters,
        'Statement' => $statement,
    ]);
}

public function updateItemByPartiQL(string $statement, array $parameters)
{
    $this->dynamoDbClient->executeStatement([
        'Statement' => $statement,
        'Parameters' => $parameters,
    ]);
}

public function deleteItemByPartiQL(string $statement, array $parameters)
{
    $this->dynamoDbClient->executeStatement([
        'Statement' => $statement,
        'Parameters' => $parameters,
    ]);
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for PHP API Reference*.

## AWS Glue examples using SDK for PHP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for PHP with AWS Glue.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3688\)](#)
- [Scenarios \(p. 3695\)](#)

## Actions

### Create a crawler

The following code example shows how to create an AWS Glue crawler.

#### SDK for PHP

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$crawlerName = "example-crawler-test-" . $uniqid;

$role = $iamService->getRole("AWSGlueServiceRole-DocExample");

$path = 's3://crawler-public-us-east-1/flight/2016/csv';
$glueService->createCrawler($crawlerName, $role['Role']['Arn'], $databaseName,
$path);

public function createCrawler($crawlerName, $role, $databaseName, $path): Result
{
    return $this->customWaiter(function () use ($crawlerName, $role, $databaseName,
$path) {
        return $this->glueClient->createCrawler([
            'Name' => $crawlerName,
            'Role' => $role,
            'DatabaseName' => $databaseName,
            'Targets' => [
                'S3Targets' =>
                [
                    [
                        'Path' => $path,
                    ]
                ],
            ],
        ]);
    });
}
```

- For API details, see [CreateCrawler](#) in *AWS SDK for PHP API Reference*.

### Create a job definition

The following code example shows how to create an AWS Glue job definition.

#### SDK for PHP

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$role = $iamService->getRole("AWSGlueServiceRole-DocExample");

$jobName = 'test-job-' . $uniqid;

$scriptLocation = "s3://$bucketName/run_job.py";
$job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);
```

```
    public function createJob($jobName, $role, $scriptLocation, $pythonVersion = '3',  
    $glueVersion = '3.0'): Result  
    {  
        return $this->glueClient->createJob([  
            'Name' => $jobName,  
            'Role' => $role,  
            'Command' => [  
                'Name' => 'glueetl',  
                'ScriptLocation' => $scriptLocation,  
                'PythonVersion' => $pythonVersion,  
            ],  
            'GlueVersion' => $glueVersion,  
        ]);  
    }  
}
```

- For API details, see [CreateJob](#) in *AWS SDK for PHP API Reference*.

## Delete a crawler

The following code example shows how to delete an AWS Glue crawler.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
echo "Delete the crawler.\n";  
$glueClient->deleteCrawler([  
    'Name' => $crawlerName,  
]);  
  
public function deleteCrawler($crawlerName)  
{  
    return $this->glueClient->deleteCrawler([  
        'Name' => $crawlerName,  
    ]);  
}
```

- For API details, see [DeleteCrawler](#) in *AWS SDK for PHP API Reference*.

## Delete a database from the Data Catalog

The following code example shows how to delete a database from the AWS Glue Data Catalog.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
echo "Delete the databases.\n";  
$glueClient->deleteDatabase([  
    'Name' => $databaseName,  
]);
```

```
public function deleteDatabase($databaseName)
{
    return $this->glueClient->deleteDatabase([
        'Name' => $databaseName,
    ]);
}
```

- For API details, see [DeleteDatabase](#) in *AWS SDK for PHP API Reference*.

## Delete a job definition

The following code example shows how to delete an AWS Glue job definition and all associated runs.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
echo "Delete the job.\n";
$glueClient->deleteJob([
    'JobName' => $job['Name'],
]);

public function deleteJob($jobName)
{
    return $this->glueClient->deleteJob([
        'JobName' => $jobName,
    ]);
}
```

- For API details, see [DeleteJob](#) in *AWS SDK for PHP API Reference*.

## Delete a table from a database

The following code example shows how to delete a table from an AWS Glue Data Catalog database.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
echo "Delete the tables.\n";
foreach ($tables['TableList'] as $table) {
    $glueService->deleteTable($table['Name'], $databaseName);
}

public function deleteTable($tableName, $databaseName)
{
    return $this->glueClient->deleteTable([
        'DatabaseName' => $databaseName,
        'Name' => $tableName,
    ]);
}
```

- For API details, see [DeleteTable](#) in *AWS SDK for PHP API Reference*.

## Get a crawler

The following code example shows how to get an AWS Glue crawler.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
echo "Waiting for crawler";
do {
    $crawler = $glueService->getCrawler($crawlerName);
    echo ".";
    sleep(10);
} while ($crawler['Crawler']['State'] != "READY");
echo "\n";

public function getCrawler($crawlerName)
{
    return $this->customWaiter(function () use ($crawlerName) {
        return $this->glueClient->getCrawler([
            'Name' => $crawlerName,
        ]);
    });
}
```

- For API details, see [GetCrawler](#) in *AWS SDK for PHP API Reference*.

## Get a database from the Data Catalog

The following code example shows how to get a database from the AWS Glue Data Catalog.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$databaseName = "doc-example-database-$uniqid";

$database = $glueService->getDatabase($databaseName);
echo "Found a database named " . $database['Database']['Name'] . "\n";

public function getDatabase(string $databaseName): Result
{
    return $this->customWaiter(function () use ($databaseName) {
        return $this->glueClient->getDatabase([
            'Name' => $databaseName,
        ]);
    });
}
```

- For API details, see [GetDatabase](#) in *AWS SDK for PHP API Reference*.

## Get a job run

The following code example shows how to get an AWS Glue job run.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$jobName = 'test-job-' . $uniqid;

$outputBucketUrl = "s3://$bucketName";
$runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

echo "waiting for job";
do {
    $jobRun = $glueService->getJobRun($jobName, $runId);
    echo ".";
    sleep(10);
} while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
echo "\n";

public function getJobRun($jobName, $runId, $predecessorsIncluded = false): Result
{
    return $this->glueClient->getJobRun([
        'JobName' => $jobName,
        'RunId' => $runId,
        'PredecessorsIncluded' => $predecessorsIncluded,
    ]);
}
```

- For API details, see [GetJobRun](#) in *AWS SDK for PHP API Reference*.

## Get runs of a job

The following code example shows how to get runs of an AWS Glue job.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$jobName = 'test-job-' . $uniqid;

$jobRuns = $glueService->getJobRuns($jobName);

public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
{
    $arguments = ['JobName' => $jobName];
    if ($maxResults) {
        $arguments['MaxResults'] = $maxResults;
    }
    if ($nextToken) {
        $arguments['NextToken'] = $nextToken;
    }
    return $this->glueClient->getJobRuns($arguments);
```

```
}
```

- For API details, see [GetJobRuns](#) in *AWS SDK for PHP API Reference*.

## Get tables from a database

The following code example shows how to get tables from a database in the AWS Glue Data Catalog.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$databaseName = "doc-example-database-$uniqid";

$tables = $glueService->getTables($databaseName);

public function getTables($databaseName): Result
{
    return $this->glueClient->getTables([
        'DatabaseName' => $databaseName,
    ]);
}
```

- For API details, see [GetTables](#) in *AWS SDK for PHP API Reference*.

## List job definitions

The following code example shows how to list AWS Glue job definitions.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$jobs = $glueService->listJobs();
echo "Current jobs:\n";
foreach ($jobs['JobNames'] as $jobsName) {
    echo "{$jobsName}\n";
}

public function listJobs($maxResults = null, $nextToken = null, $tags = []): Result
{
    $arguments = [];
    if ($maxResults) {
        $arguments['MaxResults'] = $maxResults;
    }
    if ($nextToken) {
        $arguments['NextToken'] = $nextToken;
    }
    if (!isEmpty($tags)) {
        $arguments['Tags'] = $tags;
    }
    return $this->glueClient->listJobs($arguments);
}
```

- For API details, see [ListJobs](#) in *AWS SDK for PHP API Reference*.

## Start a crawler

The following code example shows how to start an AWS Glue crawler.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$crawlerName = "example-crawler-test-" . $uniqid;  
  
$databaseName = "doc-example-database-$uniqid";  
  
$glueService->startCrawler($crawlerName);  
  
public function startCrawler($crawlerName): Result  
{  
    return $this->glueClient->startCrawler([  
        'Name' => $crawlerName,  
    ]);  
}
```

- For API details, see [StartCrawler](#) in *AWS SDK for PHP API Reference*.

## Start a job run

The following code example shows how to start an AWS Glue job run.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$jobName = 'test-job-' . $uniqid;  
  
$databaseName = "doc-example-database-$uniqid";  
  
$tables = $glueService->getTables($databaseName);  
  
$outputBucketUrl = "s3://$bucketName";  
$runId = $glueService->startJobRun($jobName, $databaseName, $tables,  
$outputBucketUrl)['JobRunId'];  
  
public function startJobRun($jobName, $databaseName, $tables, $outputBucketUrl):  
Result  
{  
    return $this->glueClient->startJobRun([  
        'JobName' => $jobName,  
        'Arguments' => [  
            'input_database' => $databaseName,  
            'input_table' => $tables['TableList'][0]['Name'],  
            'output_bucket_url' => $outputBucketUrl,
```

```
        '--input_database' => $databaseName,
        '--input_table' => $tables['TableList'][0]['Name'],
        '--output_bucket_url' => $outputBucketUrl,
    ],
]);
}
```

- For API details, see [StartJobRun](#) in *AWS SDK for PHP API Reference*.

## Scenarios

### Get started running crawlers and jobs

The following code example shows how to:

- Create and run a crawler that crawls a public Amazon Simple Storage Service (Amazon S3) bucket and generates a metadata database that describes the CSV-formatted data it finds.
- List information about databases and tables in your AWS Glue Data Catalog.
- Create and run a job that extracts CSV data from the source Amazon S3 bucket, transforms it by removing and renaming fields, and loads JSON-formatted output into another Amazon S3 bucket.
- List information about job runs and view some of the transformed data.
- Delete all resources created by the demo.

For more information, see [Tutorial: Getting started with AWS Glue Studio](#).

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace Glue;

use Aws\Glue\GlueClient;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use GuzzleHttp\Psr7\Stream;
use Iam\IamService;

class GettingStartedWithGlue
{
    public function run()
    {
        echo("-----\n");
        print("Welcome to the Amazon Glue getting started demo using PHP!\n");
        echo("-----\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $glueClient = new GlueClient($clientArgs);
        $glueService = new GlueService($glueClient);
        $iamService = new IamService();
        $crawlerName = "example-crawler-test-" . $uniqid;
```

```
AWSServiceClass::$waitTime = 5;
AWSServiceClass::$maxWaitAttempts = 20;

$role = $iamService->getRole("AWSGlueServiceRole-DocExample");

$databaseName = "doc-example-database-$uniqid";
$path = 's3://crawler-public-us-east-1/flight/2016/csv';
$glueService->createCrawler($crawlerName, $role['Role']['Arn'], $databaseName,
$path);
$glueService->startCrawler($crawlerName);

echo "Waiting for crawler";
do {
    $crawler = $glueService->getCrawler($crawlerName);
    echo ".";
    sleep(10);
} while ($crawler['Crawler']['State'] != "READY");
echo "\n";

$database = $glueService->getDatabase($databaseName);
echo "Found a database named " . $database['Database']['Name'] . "\n";

//Upload job script
$s3client = new S3Client($clientArgs);
$bucketName = "test-glue-bucket-" . $uniqid;
$s3client->createBucket([
    'Bucket' => $bucketName,
    'CreateBucketConfiguration' => ['LocationConstraint' => 'us-west-2'],
]);
$s3client->putObject([
    'Bucket' => $bucketName,
    'Key' => 'run_job.py',
    'SourceFile' => 'glue/flight_etl_job_script.py'
]);
$s3client->putObject([
    'Bucket' => $bucketName,
    'Key' => 'setup_scenario_getting_started.yaml',
    'SourceFile' => 'glue/setup_scenario_getting_started.yaml'
]);

$tables = $glueService->getTables($databaseName);

$jobName = 'test-job-' . $uniqid;
$scriptLocation = "s3://$bucketName/run_job.py";
$job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

$outputBucketUrl = "s3://$bucketName";
$runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

echo "waiting for job";
do {
    $jobRun = $glueService->getJobRun($jobName, $runId);
    echo ".";
    sleep(10);
} while (!array_intersect([$jobRun['JobRun']['JobRunState']], ['SUCCEEDED',
'STOPPED', 'FAILED', 'TIMEOUT']));
echo "\n";

$jobRuns = $glueService->getJobRuns($jobName);

$objects = $s3client->listObjects([
    'Bucket' => $bucketName,
```

```
])['Contents'];

foreach ($objects as $object) {
    echo $object['Key'] . "\n";
}

echo "Downloading " . $objects[2]['Key'] . "\n";
/** @var Stream $downloadObject */
$downloadObject = $s3client->getObject([
    'Bucket' => $bucketName,
    'Key' => $objects[2]['Key'],
])['Body']->getContents();
echo "Here is the first 1000 characters in the object.";
echo substr($downloadObject, 0, 1000);

$jobs = $glueService->listJobs();
echo "Current jobs:\n";
foreach ($jobs['JobNames'] as $jobsName) {
    echo "{$jobsName}\n";
}

echo "Delete the job.\n";
$glueClient->deleteJob([
    'JobName' => $job['Name'],
]);

echo "Delete the tables.\n";
foreach ($tables['TableList'] as $table) {
    $glueService->deleteTable($table['Name'], $databaseName);
}

echo "Delete the databases.\n";
$glueClient->deleteDatabase([
    'Name' => $databaseName,
]);

echo "Delete the crawler.\n";
$glueClient->deleteCrawler([
    'Name' => $crawlerName,
]);

$deleteObjects = $s3client->listObjectsV2([
    'Bucket' => $bucketName,
]);
echo "Delete all objects in the bucket.\n";
$deleteObjects = $s3client->deleteObjects([
    'Bucket' => $bucketName,
    'Delete' => [
        'Objects' => $deleteObjects['Contents'],
    ]
]);
echo "Delete the bucket.\n";
$s3client->deleteBucket(['Bucket' => $bucketName]);

echo "This job was brought to you by the number $unqid\n";
}

namespace Glue;

use Aws\Glue\GlueClient;
use Aws\Result;

use function PHPUnit\Framework\isEmpty;

class GlueService extends \AwsUtilities\AWSServiceClass
```

```
{  
    protected GlueClient $glueClient;  
  
    public function __construct($glueClient)  
    {  
        $this->glueClient = $glueClient;  
    }  
  
    public function getCrawler($crawlerName)  
    {  
        return $this->customWaiter(function () use ($crawlerName) {  
            return $this->glueClient->getCrawler([  
                'Name' => $crawlerName,  
            ]);  
        });  
    }  
  
    public function createCrawler($crawlerName, $role, $databaseName, $path): Result  
    {  
        return $this->customWaiter(function () use ($crawlerName, $role, $databaseName,  
$path) {  
            return $this->glueClient->createCrawler([  
                'Name' => $crawlerName,  
                'Role' => $role,  
                'DatabaseName' => $databaseName,  
                'Targets' => [  
                    'S3Targets' =>  
                    [[  
                        'Path' => $path,  
                    ]]  
                ],  
            ]);  
        });  
    }  
  
    public function startCrawler($crawlerName): Result  
    {  
        return $this->glueClient->startCrawler([  
            'Name' => $crawlerName,  
        ]);  
    }  
  
    public function getDatabase(string $databaseName): Result  
    {  
        return $this->customWaiter(function () use ($databaseName) {  
            return $this->glueClient->getDatabase([  
                'Name' => $databaseName,  
            ]);  
        });  
    }  
  
    public function getTables($databaseName): Result  
    {  
        return $this->glueClient->getTables([  
            'DatabaseName' => $databaseName,  
        ]);  
    }  
  
    public function createJob($jobName, $role, $scriptLocation, $pythonVersion = '3',  
$glueVersion = '3.0'): Result  
    {  
        return $this->glueClient->createJob([  
            'Name' => $jobName,  
            'Role' => $role,  
            'Command' => [  
                'Name' => 'glueetl',  
            ]  
        ]);  
    }  
}
```

```
        'ScriptLocation' => $scriptLocation,
        'PythonVersion' => $pythonVersion,
    ],
    'GlueVersion' => $glueVersion,
]);
}

public function startJobRun($jobName, $databaseName, $tables, $outputBucketUrl): Result
{
    return $this->glueClient->startJobRun([
        'JobName' => $jobName,
        'Arguments' => [
            'input_database' => $databaseName,
            'input_table' => $tables['TableList'][0]['Name'],
            'output_bucket_url' => $outputBucketUrl,
            '--input_database' => $databaseName,
            '--input_table' => $tables['TableList'][0]['Name'],
            '--output_bucket_url' => $outputBucketUrl,
        ],
    ]);
}

public function listJobs($maxResults = null, $nextToken = null, $tags = []): Result
{
    $arguments = [];
    if ($maxResults) {
        $arguments['MaxResults'] = $maxResults;
    }
    if ($nextToken) {
        $arguments['NextToken'] = $nextToken;
    }
    if (!isEmpty($tags)) {
        $arguments['Tags'] = $tags;
    }
    return $this->glueClient->listJobs($arguments);
}

public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''): Result
{
    $arguments = ['JobName' => $jobName];
    if ($maxResults) {
        $arguments['MaxResults'] = $maxResults;
    }
    if ($nextToken) {
        $arguments['NextToken'] = $nextToken;
    }
    return $this->glueClient->getJobRuns($arguments);
}

public function getJobRun($jobName, $runId, $predecessorsIncluded = false): Result
{
    return $this->glueClient->getJobRun([
        'JobName' => $jobName,
        'RunId' => $runId,
        'PredecessorsIncluded' => $predecessorsIncluded,
    ]);
}

public function deleteJob($jobName)
{
    return $this->glueClient->deleteJob([
        'JobName' => $jobName,
    ]);
}
```

```
public function deleteTable($tableName, $databaseName)
{
    return $this->glueClient->deleteTable([
        'DatabaseName' => $databaseName,
        'Name' => $tableName,
    ]);
}

public function deleteDatabase($databaseName)
{
    return $this->glueClient->deleteDatabase([
        'Name' => $databaseName,
    ]);
}

public function deleteCrawler($crawlerName)
{
    return $this->glueClient->deleteCrawler([
        'Name' => $crawlerName,
    ]);
}
```

- For API details, see the following topics in *AWS SDK for PHP API Reference*.
  - [CreateCrawler](#)
  - [CreateJob](#)
  - [DeleteCrawler](#)
  - [DeleteDatabase](#)
  - [DeleteJob](#)
  - [DeleteTable](#)
  - [GetCrawler](#)
  - [GetDatabase](#)
  - [GetDatabases](#)
  - [GetJob](#)
  - [GetJobRun](#)
  - [GetJobRuns](#)
  - [GetTables](#)
  - [ListJobs](#)
  - [StartCrawler](#)
  - [StartJobRun](#)

## IAM examples using SDK for PHP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for PHP with AWS Identity and Access Management.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3701\)](#)
- [Scenarios \(p. 3709\)](#)

## Actions

### Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

#### SDK for PHP

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

$assumeRolePolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Principal\": \"{\\\"AWS\\\": \\"${user['Arn']}\\\"}\",
            \"Action\": \"sts:AssumeRole\"
        }
    ]
}";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
    $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": \"s3>ListAllMyBuckets\",
            \"Resource\": \"arn:aws:s3:::*\\\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
    $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'], $listAllBucketsPolicy['Arn']);

    public function attachRolePolicy($roleName, $policyArn)
    {
        return $this->customWaiter(function () use ($roleName, $policyArn) {
            $this->iamClient->attachRolePolicy([
                'PolicyArn' => $policyArn,
                'RoleName' => $roleName,
            ]);
        });
    }
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for PHP API Reference*.

## Create a policy

The following code example shows how to create an IAM policy.

#### SDK for PHP

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

$listAllBucketsPolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": \"s3>ListAllMyBuckets\",
            \"Resource\": \"arn:aws:s3:::*\"}
    ]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
    $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

public function createPolicy(string $policyName, string $policyDocument)
{
    $result = $this->customWaiter(function () use ($policyName, $policyDocument) {
        return $this->iamClient->createPolicy([
            'PolicyName' => $policyName,
            'PolicyDocument' => $policyDocument,
        ]);
    });
    return $result['Policy'];
}
```

- For API details, see [CreatePolicy](#) in *AWS SDK for PHP API Reference*.

## Create a role

The following code example shows how to create an IAM role.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

$assumeRolePolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Principal\": \"$user[\"Arn\"]\",
            \"Action\": \"sts:AssumeRole\"
        }
    ]
}";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
    $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

/**
 * @param string $roleName
 * @param string $rolePolicyDocument
 * @return array
 * @throws AwsException
 */
public function createRole(string $roleName, string $rolePolicyDocument)
{
```

```
    $result = $this->customWaiter(function () use ($roleName, $rolePolicyDocument)
{
    return $this->iamClient->createRole([
        'AssumeRolePolicyDocument' => $rolePolicyDocument,
        'RoleName' => $roleName,
    ]);
});
return $result['Role'];
}
```

- For API details, see [CreateRole](#) in *AWS SDK for PHP API Reference*.

## Create a service-linked role

The following code example shows how to create an IAM service-linked role.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

public function createServiceLinkedRole($awsServiceName, $customSuffix = "",
$description = "")
{
    $createServiceLinkedRoleArguments = ['AWSServiceName' => $awsServiceName];
    if ($customSuffix) {
        $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
    }
    if ($description) {
        $createServiceLinkedRoleArguments['Description'] = $description;
    }
    return $this->iamClient-
>createServiceLinkedRole($createServiceLinkedRoleArguments);
}
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for PHP API Reference*.

## Create a user

The following code example shows how to create an IAM user.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";
```

```
/**  
 * @param string $name  
 * @return array  
 * @throws AwsException  
 */  
public function createUser(string $name): array  
{  
    $result = $this->iamClient->createUser([  
        'UserName' => $name,  
    ]);  
  
    return $result['User'];  
}
```

- For API details, see [CreateUser](#) in *AWS SDK for PHP API Reference*.

## Get a policy

The following code example shows how to get an IAM policy.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();  
$service = new IamService();  
  
public function getPolicy($policyArn)  
{  
    return $this->customWaiter(function () use ($policyArn) {  
        return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);  
    });  
}
```

- For API details, see [GetPolicy](#) in *AWS SDK for PHP API Reference*.

## Get a role

The following code example shows how to get an IAM role.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();  
$service = new IamService();  
  
public function getRole($roleName)  
{  
    return $this->customWaiter(function () use ($roleName) {  
        return $this->iamClient->getRole(['RoleName' => $roleName]);  
    });  
}
```

- For API details, see [GetRole](#) in *AWS SDK for PHP API Reference*.

## Get the account password policy

The following code example shows how to get the IAM account password policy.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

    public function getAccountPasswordPolicy()
    {
        return $this->iamClient->getAccountPasswordPolicy();
    }
```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for PHP API Reference*.

## List SAML providers

The following code example shows how to list SAML providers for IAM.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

    public function listSAMLProviders()
    {
        return $this->iamClient->listSAMLProviders();
    }
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for PHP API Reference*.

## List groups

The following code example shows how to list IAM groups.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
```

```
$service = new IamService();

    public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listGroupsArguments = [];
        if ($pathPrefix) {
            $listGroupsArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listGroupsArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listGroupsArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listGroups($listGroupsArguments);
    }
}
```

- For API details, see [ListGroups](#) in *AWS SDK for PHP API Reference*.

## List inline policies for a role

The following code example shows how to list inline policies for an IAM role.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

    public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
    {
        $listRolePoliciesArguments = ['RoleName' => $roleName];
        if ($marker) {
            $listRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->customWaiter(function () use ($listRolePoliciesArguments) {
            return $this->iamClient->listRolePolicies($listRolePoliciesArguments);
        });
    }
}
```

- For API details, see [ListRolePolicies](#) in *AWS SDK for PHP API Reference*.

## List policies

The following code example shows how to list IAM policies.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

    public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listPoliciesArguments = [];
        if ($pathPrefix) {
            $listPoliciesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listPoliciesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listPoliciesArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listPolicies($listPoliciesArguments);
    }
```

- For API details, see [ListPolicies](#) in *AWS SDK for PHP API Reference*.

## List policies attached to a role

The following code example shows how to list policies attached to an IAM role.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

    public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listAttachRolePoliciesArguments = ['RoleName' => $roleName];
        if ($pathPrefix) {
            $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;
        }
        if ($marker) {
            $listAttachRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->iamClient-
>listAttachedRolePolicies($listAttachRolePoliciesArguments);
    }
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for PHP API Reference*.

## List roles

The following code example shows how to list IAM roles.

## SDK for PHP

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

/**
 * @param string $pathPrefix
 * @param string $marker
 * @param int $maxItems
 * @return Result
 * $roles = $service->listRoles();
 */
public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
{
    $listRolesArguments = [];
    if ($pathPrefix) {
        $listRolesArguments["PathPrefix"] = $pathPrefix;
    }
    if ($marker) {
        $listRolesArguments["Marker"] = $marker;
    }
    if ($maxItems) {
        $listRolesArguments["MaxItems"] = $maxItems;
    }
    return $this->iamClient->listRoles($listRolesArguments);
}
```

- For API details, see [ListRoles](#) in *AWS SDK for PHP API Reference*.

## List users

The following code example shows how to list IAM users.

## SDK for PHP

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$uuid = uniqid();
$service = new IamService();

public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)
{
    $listUsersArguments = [];
    if ($pathPrefix) {
        $listUsersArguments["PathPrefix"] = $pathPrefix;
    }
    if ($marker) {
        $listUsersArguments["Marker"] = $marker;
    }
    if ($maxItems) {
        $listUsersArguments["MaxItems"] = $maxItems;
    }

    return $this->iamClient->listUsers($listUsersArguments);
```

```
}
```

- For API details, see [ListUsers](#) in *AWS SDK for PHP API Reference*.

## Scenarios

### Create a user and assume a role

The following code example shows how to:

- Create a user who has no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list Amazon S3 buckets using temporary credentials.
- Delete the policy, role, and user.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IamService;

echo("-----\n");
print("Welcome to the Amazon IAM getting started demo using PHP!\n");
echo("-----\n");
$uuid = uniqid();
 getService()];

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{$
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "{$AWS": "{$user['Arn']}",
            "Action": "sts:AssumeRole"
        }
    ]
}";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
    $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{$
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",

```

```

        \\"Action\\": \"s3>ListAllMyBuckets\",
        \\"Resource\\": \"arn:aws:s3:::*\"]]
};

$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
    $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'], $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
    \\"Version\\": \"2012-10-17\",
    \\"Statement\\": [
        {
            \\"Effect\\": \"Allow\",
            \\"Action\\": \"sts:AssumeRole\",
            \\"Resource\\": \"{$assumeRoleRole['Arn']}\"]
    ];
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
    $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest', 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
    'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest', 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should now run!\\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'], $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\\n";

```

- For API details, see the following topics in *AWS SDK for PHP API Reference*.
  - [AttachRolePolicy](#)

- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## Lambda examples using SDK for PHP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for PHP with AWS Lambda.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3711\)](#)
- [Scenarios \(p. 3714\)](#)

## Actions

### Create a function

The following code example shows how to create a Lambda function.

#### SDK for PHP

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function createFunction($functionName, $role, $bucketName, $handler)
{
    //This assumes the Lambda function is in an S3 bucket.
    return $this->customWaiter(function () use ($functionName, $role, $bucketName,
$handler) {
        return $this->lambdaClient->createFunction([
            'Code' => [
                'S3Bucket' => $bucketName,
                'S3Key' => $functionName,
            ],
            'FunctionName' => $functionName,
            'Role' => $role['Arn'],
            'Runtime' => 'python3.9',
            'Handler' => "$handler.lambda_handler",
        ]);
    });
}
```

- For API details, see [CreateFunction](#) in *AWS SDK for PHP API Reference*.

## Delete a function

The following code example shows how to delete a Lambda function.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function deleteFunction($functionName)
{
    return $this->lambdaClient->deleteFunction([
        'FunctionName' => $functionName,
    ]);
}
```

- For API details, see [DeleteFunction](#) in *AWS SDK for PHP API Reference*.

## Get a function

The following code example shows how to get a Lambda function.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function getFunction($functionName)
{
    return $this->lambdaClient->getFunction([
        'FunctionName' => $functionName,
    ]);
}
```

- For API details, see [GetFunction](#) in *AWS SDK for PHP API Reference*.

## Invoke a function

The following code example shows how to invoke a Lambda function.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function invoke($functionName, $params, $logType = 'None')
```

```
        return $this->lambdaClient->invoke([
            'FunctionName' => $functionName,
            'Payload' => json_encode($params),
            'LogType' => $logType,
        ]);
    }
```

- For API details, see [Invoke](#) in *AWS SDK for PHP API Reference*.

## List functions

The following code example shows how to list Lambda functions.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function listFunctions($maxItems = 50, $marker = null)
{
    if (is_null($marker)) {
        return $this->lambdaClient->listFunctions([
            'MaxItems' => $maxItems,
        ]);
    }

    return $this->lambdaClient->listFunctions([
        'Marker' => $marker,
        'MaxItems' => $maxItems,
    ]);
}
```

- For API details, see [ListFunctions](#) in *AWS SDK for PHP API Reference*.

## Update function code

The following code example shows how to update Lambda function code.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function updateFunctionCode($functionName, $s3Bucket, $s3Key)
{
    return $this->lambdaClient->updateFunctionCode([
        'FunctionName' => $functionName,
        'S3Bucket' => $s3Bucket,
        'S3Key' => $s3Key,
    ]);
}
```

- For API details, see [UpdateFunctionCode](#) in *AWS SDK for PHP API Reference*.

## Update function configuration

The following code example shows how to update Lambda function configuration.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public function updateFunctionConfiguration($functionName, $handler, $environment = '') {
    {
        return $this->lambdaClient->updateFunctionConfiguration([
            'FunctionName' => $functionName,
            'Handler' => "$handler.lambda_handler",
            'Environment' => $environment,
        ]);
    }
}
```

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for PHP API Reference*.

## Scenarios

### Get started with functions

The following code example shows how to:

- Create an AWS Identity and Access Management (IAM) role that grants Lambda permission to write to logs.
- Create a Lambda function and upload handler code.
- Invoke the function with a single parameter and get results.
- Update the function code and configure its Lambda environment with an environment variable.
- Invoke the function with new parameters and get results. Display the execution log that's returned from the invocation.
- List the functions for your account.
- Delete the IAM role and the Lambda function.

For more information, see [Create a Lambda function with the console](#).

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
namespace Lambda;

use Aws\S3\S3Client;
use GuzzleHttp\Psr7\Stream;
use Iam\IamService;

class GettingStartedWithLambda
{
    public function run()
    {
```

```
echo("-----\n");
print("Welcome to the AWS Lambda getting started demo using PHP!\n");
echo("-----\n");

$clientArgs = [
    'region' => 'us-west-2',
    'version' => 'latest',
    'profile' => 'default',
];
$uniqid = uniqid();

$iامService = new IamService();
$s3client = new S3Client($clientArgs);
$lambdaService = new LambdaService();

echo "First, let's create a role to run our Lambda code.\n";
$roleName = "test-lambda-role-$uniqid";
$rolePolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Principal\": {
                \"Service\": \"lambda.amazonaws.com\"
            },
            \"Action\": \"sts:AssumeRole\"
        }
    ]
}";
$role = $iamService->createRole($roleName, $rolePolicyDocument);
echo "Created role {$role['RoleName']}.\n";

$iamService->attachRolePolicy(
    $role['RoleName'],
    "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
);
echo "Attached the AWSLambdaBasicExecutionRole to {$role['RoleName']}.\n";

echo "\nNow let's create an S3 bucket and upload our Lambda code there.\n";
$bucketName = "test-example-bucket-$uniqid";
$s3client->createBucket([
    'Bucket' => $bucketName,
]);
echo "Created bucket $bucketName.\n";

$functionName = "doc_example_lambda_$uniqid";
$codeBasic = "lambda_handler_basic.zip";
$handler = "lambda_handler_basic";
$file = file_get_contents($codeBasic);
$s3client->putObject([
    'Bucket' => $bucketName,
    'Key' => $functionName,
    'Body' => $file,
]);
echo "Uploaded the Lambda code.\n";

$createLambdaFunction = $lambdaService->createFunction($functionName, $role,
$bucketName, $handler);
// Wait until the function has finished being created.
do {
    $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
} while ($getLambdaFunction['Configuration']['State'] == "Pending");
echo "Created Lambda function {$getLambdaFunction['Configuration']
['FunctionName']}.\n";
```

```
sleep(1);

echo "\nOk, let's invoke that Lambda code.\n";
$basicParams = [
    'action' => 'increment',
    'number' => 3,
];
/** @var Stream $invokeFunction */
$invokeFunction = $lambdaService->invoke($functionName, $basicParams)
['Payload'];
$result = json_decode($invokeFunction->getContents())->result;
echo "After invoking the Lambda code with the input of {$basicParams['number']} we received $result.\n";

echo "\nSince that's working, let's update the Lambda code.\n";
$codeCalculator = "lambda_handler_calculator.zip";
$handlerCalculator = "lambda_handler_calculator";
echo "First, put the new code into the S3 bucket.\n";
$file = file_get_contents($codeCalculator);
$s3client->putObject([
    'Bucket' => $bucketName,
    'Key' => $functionName,
    'Body' => $file,
]);
echo "New code uploaded.\n";

$lambdaService->updateFunctionCode($functionName, $bucketName, $functionName);
// Wait for the Lambda code to finish updating.
do {
    $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
} while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
"Successful");
echo "New Lambda code uploaded.\n";

$environment = [
    'Variable' => ['Variables' => ['LOG_LEVEL' => 'DEBUG']],
];
$lambdaService->updateFunctionConfiguration($functionName, $handlerCalculator,
$environment);
do {
    $getLambdaFunction = $lambdaService-
>getFunction($createLambdaFunction['FunctionName']);
} while ($getLambdaFunction['Configuration']['LastUpdateStatus'] !==
"Successful");
echo "Lambda code updated with new handler and a LOG_LEVEL of DEBUG for more
information.\n";

echo "Invoke the new code with some new data.\n";
$calculatorParams = [
    'action' => 'plus',
    'x' => 5,
    'y' => 4,
];
$invokeFunction = $lambdaService->invoke($functionName, $calculatorParams,
"Tail");
$result = json_decode($invokeFunction['Payload']->getContents())->result;
echo "Indeed, {$calculatorParams['x']} + {$calculatorParams['y']} does equal
$result.\n";
echo "Here's the extra debug info: ";
echo base64_decode($invokeFunction['LogResult']) . "\n";

echo "\nBut what happens if you try to divide by zero?\n";
$divZeroParams = [
    'action' => 'divide',
    'x' => 5,
```

```

        'y' => 0,
    ];
    $invokeFunction = $lambdaService->invoke($functionName, $divZeroParams,
"Tail");
    $result = json_decode($invokeFunction['Payload']->getContents())->result;
    echo "You get a |$result| result.\n";
    echo "And an error message: ";
    echo base64_decode($invokeFunction['LogResult']) . "\n";

    echo "\nHere's all the Lambda functions you have in this Region:\n";
    $listLambdaFunctions = $lambdaService->listFunctions(5);
    $allLambdaFunctions = $listLambdaFunctions['Functions'];
    $next = $listLambdaFunctions->get('NextMarker');
    while ($next != false) {
        $listLambdaFunctions = $lambdaService->listFunctions(5, $next);
        $next = $listLambdaFunctions->get('NextMarker');
        $allLambdaFunctions = array_merge($allLambdaFunctions,
$listLambdaFunctions['Functions']);
    }
    foreach ($allLambdaFunctions as $function) {
        echo "{$function['FunctionName']} \n";
    }

    echo "\n\nAnd don't forget to clean up your data!\n";

    $lambdaService->deleteFunction($functionName);
    echo "Deleted Lambda function.\n";
    $iamService->deleteRole($role['RoleName']);
    echo "Deleted Role.\n";
    $deleteObjects = $s3client->listObjectsV2([
        'Bucket' => $bucketName,
    ]);
    $deleteObjects = $s3client->deleteObjects([
        'Bucket' => $bucketName,
        'Delete' => [
            'Objects' => $deleteObjects['Contents'],
        ]
    ]);
    echo "Deleted all objects from the S3 bucket.\n";
    $s3client->deleteBucket(['Bucket' => $bucketName]);
    echo "Deleted the bucket.\n";
}
}

```

- For API details, see the following topics in *AWS SDK for PHP API Reference*.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## Amazon S3 examples using SDK for PHP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for PHP with Amazon Simple Storage Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3718\)](#)
- [Scenarios \(p. 3721\)](#)

## Actions

### Copy an object from one bucket to another

The following code example shows how to copy an S3 object from one bucket to another.

#### SDK for PHP

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Simple copy of an object.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);

try {
    $folder = "copied-folder";
    $s3client->copyObject([
        'Bucket' => $bucket_name,
        'CopySource' => "$bucket_name/$file_name",
        'Key' => "$folder/$file_name-copy",
    ]);
    echo "Copied $file_name to $folder/$file_name-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $file_name with error: " . $exception->getMessage();
    exit("Please fix error with object copying before continuing.");
}
```

- For API details, see [CopyObject](#) in *AWS SDK for PHP API Reference*.

## Create a bucket

The following code example shows how to create an S3 bucket.

#### SDK for PHP

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);

try {
    $s3client->createBucket([
        'Bucket' => $bucket_name,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
    echo "Created bucket named: $bucket_name \n";
} catch (Exception $exception) {
```

```
    echo "Failed to create bucket $bucket_name with error: " . $exception->getMessage();
    exit("Please fix error with bucket creation before continuing.");
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for PHP API Reference*.

## Delete an empty bucket

The following code example shows how to delete an empty S3 bucket.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an empty bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);

try {
    $s3client->deleteBucket([
        'Bucket' => $bucket_name,
    ]);
    echo "Deleted bucket $bucket_name.\n";
} catch (Exception $exception) {
    echo "Failed to delete $bucket_name with error: " . $exception->getMessage();
    exit("Please fix error with bucket deletion before continuing.");
}
```

- For API details, see [DeleteBucket](#) in *AWS SDK for PHP API Reference*.

## Delete multiple objects

The following code example shows how to delete multiple objects from an S3 bucket.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete a set of objects from a list of keys.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);

try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [
            'Key' => $content['Key'],
        ];
    }
    $s3client->deleteObjects([
        'Bucket' => $bucket_name,
        'Key' => $file_name,
        'Delete' => [
            'Objects' => $objects,
        ],
    ],
}
```

```
]);
$check = $s3client->listObjects([
    'Bucket' => $bucket_name,
]);
if (count($check) <= 0) {
    throw new Exception("Bucket wasn't empty.");
}
echo "Deleted all objects and folders from $bucket_name.\n";
} catch (Exception $exception) {
    echo "Failed to delete $file_name from $bucket_name with error: " . $exception-
>getMessage();
    exit("Please fix error with object deletion before continuing.");
}
```

- For API details, see [DeleteObjects](#) in *AWS SDK for PHP API Reference*.

## Get an object from a bucket

The following code example shows how to read data from an object in an S3 bucket.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get an object.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);

try {
    $file = $s3client->getObject([
        'Bucket' => $bucket_name,
        'Key' => $file_name,
    ]);
    $body = $file->get('Body');
    $body->rewind();
    echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
    echo "Failed to download $file_name from $bucket_name with error: " . $exception-
>getMessage();
    exit("Please fix error with file downloading before continuing.");
}
```

- For API details, see [GetObject](#) in *AWS SDK for PHP API Reference*.

## List objects in a bucket

The following code example shows how to list objects in an S3 bucket.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List objects in a bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);
```

```
try {
    $contents = $s3client->listObjects([
        'Bucket' => $bucket_name,
    ]);
    echo "The contents of your bucket are: \n";
    foreach ($contents['Contents'] as $content) {
        echo $content['Key'] . "\n";
    }
} catch (Exception $exception) {
    echo "Failed to list objects in $bucket_name with error: " . $exception-
>getMessage();
    exit("Please fix error with listing objects before continuing.");
}
```

- For API details, see [ListObjects](#) in *AWS SDK for PHP API Reference*.

## Upload an object to a bucket

The following code example shows how to upload an object to an S3 bucket.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Upload an object to a bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);

$file_name = "local-file-" . uniqid();
try {
    $s3client->putObject([
        'Bucket' => $bucket_name,
        'Key' => $file_name,
        'SourceFile' => 'testfile.txt'
    ]);
    echo "Uploaded $file_name to $bucket_name.\n";
} catch (Exception $exception) {
    echo "Failed to upload $file_name with error: " . $exception->getMessage();
    exit("Please fix error with file upload before continuing.");
}
```

- For API details, see [PutObject](#) in *AWS SDK for PHP API Reference*.

## Scenarios

### Get started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.

- Delete a bucket.

## SDK for PHP

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\S3\S3Client;

echo("-----\n");
print("Welcome to the Amazon S3 getting started demo using PHP!\n");
echo("-----\n");

$region = 'us-west-2';
$version = 'latest';

$s3client = new S3Client([
    'region' => $region,
    'version' => $version
]);
/* Inline declaration example
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2', 'version' => 'latest']);
*/

$bucket_name = "doc-example-bucket-" . uniqid();

try {
    $s3client->createBucket([
        'Bucket' => $bucket_name,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
    echo "Created bucket named: $bucket_name \n";
} catch (Exception $exception) {
    echo "Failed to create bucket $bucket_name with error: " . $exception-
>getMessage();
    exit("Please fix error with bucket creation before continuing.");
}

$file_name = "local-file-" . uniqid();
try {
    $s3client->putObject([
        'Bucket' => $bucket_name,
        'Key' => $file_name,
        'SourceFile' => 'testfile.txt'
    ]);
    echo "Uploaded $file_name to $bucket_name.\n";
} catch (Exception $exception) {
    echo "Failed to upload $file_name with error: " . $exception->getMessage();
    exit("Please fix error with file upload before continuing.");
}

try {
    $file = $s3client->getObject([
        'Bucket' => $bucket_name,
        'Key' => $file_name,
    ]);
    $body = $file->get('Body');
    $body->rewind();
    echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
```

```
    echo "Failed to download $file_name from $bucket_name with error: " . $exception->getMessage();
    exit("Please fix error with file downloading before continuing.");
}

try {
    $folder = "copied-folder";
    $s3client->copyObject([
        'Bucket' => $bucket_name,
        'CopySource' => "$bucket_name/$file_name",
        'Key' => "$folder/$file_name-copy",
    ]);
    echo "Copied $file_name to $folder/$file_name-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $file_name with error: " . $exception->getMessage();
    exit("Please fix error with object copying before continuing.");
}

try {
    $contents = $s3client->listObjects([
        'Bucket' => $bucket_name,
    ]);
    echo "The contents of your bucket are: \n";
    foreach ($contents['Contents'] as $content) {
        echo $content['Key'] . "\n";
    }
} catch (Exception $exception) {
    echo "Failed to list objects in $bucket_name with error: " . $exception->getMessage();
    exit("Please fix error with listing objects before continuing.");
}

try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [
            'Key' => $content['Key'],
        ];
    }
    $s3client->deleteObjects([
        'Bucket' => $bucket_name,
        'Key' => $file_name,
        'Delete' => [
            'Objects' => $objects,
        ],
    ]);
    $check = $s3client->listObjects([
        'Bucket' => $bucket_name,
    ]);
    if (count($check) <= 0) {
        throw new Exception("Bucket wasn't empty.");
    }
    echo "Deleted all objects and folders from $bucket_name.\n";
} catch (Exception $exception) {
    echo "Failed to delete $file_name from $bucket_name with error: " . $exception->getMessage();
    exit("Please fix error with object deletion before continuing.");
}

try {
    $s3client->deleteBucket([
        'Bucket' => $bucket_name,
    ]);
    echo "Deleted bucket $bucket_name.\n";
} catch (Exception $exception) {
    echo "Failed to delete $bucket_name with error: " . $exception->getMessage();
```

```
        exit("Please fix error with bucket deletion before continuing.");
}

echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- For API details, see the following topics in *AWS SDK for PHP API Reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

## Amazon SNS examples using SDK for PHP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for PHP with Amazon Simple Notification Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3724\)](#)

## Actions

### Check whether a phone number is opted out

The following code example shows how to check whether a phone number is opted out of receiving Amazon SNS messages.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Indicates whether the phone number owner has opted out of receiving SMS messages
 * from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnsClient = new SnsClient([
    'profile' => 'default',
```

```
        'region' => 'us-east-1',
        'version' => '2010-03-31'
    ]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [CheckIfPhoneNumberIsOptedOut](#) in [AWS SDK for PHP API Reference](#).

### Confirm an endpoint owner wants to receive messages

The following code example shows how to confirm the owner of an endpoint wants to receive Amazon SNS messages by validating the token sent to the endpoint by an earlier Subscribe action.

#### SDK for PHP

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Verifies an endpoint owner's intent to receive messages by validating the token sent
 * to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
// output error message if fails
error_log($e->getMessage());
}
```

- For API details, see [ConfirmSubscription](#) in *AWS SDK for PHP API Reference*.

## Create a topic

The following code example shows how to create an Amazon SNS topic.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Create a Simple Notification Service topics in your AWS account at the requested
region.
*
* This code expects that you have AWS credentials set up per:
* https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
*/
$SnsClient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$topicname = 'myTopic';

try {
    $result = $SnsClient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [CreateTopic](#) in *AWS SDK for PHP API Reference*.

## Delete a subscription

The following code example shows how to delete an Amazon SNS subscription.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [Unsubscribe](#) in [AWS SDK for PHP API Reference](#).

## Delete a topic

The following code example shows how to delete an Amazon SNS topic and all subscriptions to that topic.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Deletes a SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
```

```
]);
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For API details, see [DeleteTopic](#) in *AWS SDK for PHP API Reference*.

## Get the properties of a topic

The following code example shows how to get the properties of an Amazon SNS topic.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for PHP API Reference*.

## Get the settings for sending SMS messages

The following code example shows how to get the settings for sending Amazon SNS SMS messages.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';
```

```
use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Get the type of SMS Message sent by default from the AWS SNS service.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [GetSMSAttributes](#) in [AWS SDK for PHP API Reference](#).

## List opted out phone numbers

The following code example shows how to list phone numbers that are opted out of receiving Amazon SNS messages.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages from
 * your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut([
```

```
]);
var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [ListPhoneNumbersOptedOut](#) in [AWS SDK for PHP API Reference](#).

## List the subscribers of a topic

The following code example shows how to retrieve the list of subscribers of an Amazon SNS topic.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listSubscriptions([
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For API details, see [ListSubscriptions](#) in [AWS SDK for PHP API Reference](#).

## List topics

The following code example shows how to list Amazon SNS topics.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Returns a list of the requester's topics from your AWS SNS account in the region
 * specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listTopics([
        []
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For API details, see [ListTopics](#) in *AWS SDK for PHP API Reference*.

## Publish an SMS text message

The following code example shows how to publish SMS messages using Amazon SNS.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Sends a a text message (SMS message) directly to a phone number using Amazon SNS.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';
```

```
try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [Publish](#) in [AWS SDK for PHP API Reference](#).

## Publish to a topic

The following code example shows how to publish messages to an Amazon SNS topic.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [Publish](#) in [AWS SDK for PHP API Reference](#).

## Set the default settings for sending SMS messages

The following code example shows how to set the default settings for sending SMS messages using Amazon SNS.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
$SnsClient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnsClient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For more information, see [AWS SDK for PHP Developer Guide](#).
- For API details, see [SetSmsAttributes](#) in [AWS SDK for PHP API Reference](#).

## Set topic attributes

The following code example shows how to set Amazon SNS topic attributes.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnsClient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
```

```
        'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSClient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For API details, see [SetTopicAttributes](#) in *AWS SDK for PHP API Reference*.

## Subscribe an HTTP endpoint to a topic

The following code example shows how to subscribe an HTTP or HTTPS endpoint so it receives notifications from an Amazon SNS topic.

### SDK for PHP

#### Note

There's more on [GitHub](#). Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSClient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSClient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
}
```

```
    } catch (AwsException $e) {
        // output error message if fails
        error_log($e->getMessage());
    }
}
```

- For API details, see [Subscribe](#) in *AWS SDK for PHP API Reference*.

## Subscribe an email address to a topic

The following code example shows how to subscribe an email address to an Amazon SNS topic.

### SDK for PHP

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require 'vendor/autoload.php';

use Aws\Sns\SnsClient;
use Aws\Exception\AwsException;

/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- For API details, see [Subscribe](#) in *AWS SDK for PHP API Reference*.

## Cross-service examples using SDK for PHP

The following sample applications use the AWS SDK for PHP to work across multiple AWS services.

### Examples

- [Create an Aurora Serverless work item tracker \(p. 3736\)](#)

## Create an Aurora Serverless work item tracker

### SDK for PHP

Shows how to use the AWS SDK for PHP to create a web application that tracks work items in an Amazon RDS database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses a front end built with React.js to interact with a RESTful PHP backend.

- Integrate a React.js web application with AWS services.
- List, add, update, and delete items in an Amazon RDS table.
- Send an email report of filtered work items using Amazon SES.
- Deploy and manage example resources with the included AWS CloudFormation script.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

## Code examples for SDK for Python (Boto3)

The code examples in this topic show you how to use the AWS SDK for Python (Boto3) with AWS.

### Examples

- [Single-service actions and scenarios using SDK for Python \(Boto3\) \(p. 3736\)](#)
- [Cross-service examples using SDK for Python \(Boto3\) \(p. 4304\)](#)

## Single-service actions and scenarios using SDK for Python (Boto3)

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with AWS services.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Services

- [ACM examples using SDK for Python \(Boto3\) \(p. 3737\)](#)
- [API Gateway examples using SDK for Python \(Boto3\) \(p. 3751\)](#)
- [Application Recovery Controller examples using SDK for Python \(Boto3\) \(p. 3766\)](#)

- [Audit Manager examples using SDK for Python \(Boto3\) \(p. 3768\)](#)
- [Aurora examples using SDK for Python \(Boto3\) \(p. 3775\)](#)
- [CloudFront examples using SDK for Python \(Boto3\) \(p. 3799\)](#)
- [CloudWatch examples using SDK for Python \(Boto3\) \(p. 3801\)](#)
- [Amazon Cognito Identity Provider examples using SDK for Python \(Boto3\) \(p. 3813\)](#)
- [Amazon Comprehend examples using SDK for Python \(Boto3\) \(p. 3829\)](#)
- [AWS Config examples using SDK for Python \(Boto3\) \(p. 3853\)](#)
- [Device Farm examples using SDK for Python \(Boto3\) \(p. 3856\)](#)
- [DynamoDB examples using SDK for Python \(Boto3\) \(p. 3862\)](#)
- [Amazon EC2 examples using SDK for Python \(Boto3\) \(p. 3894\)](#)
- [Amazon EC2 Auto Scaling examples using SDK for Python \(Boto3\) \(p. 3923\)](#)
- [Amazon EMR examples using SDK for Python \(Boto3\) \(p. 3937\)](#)
- [AWS Glue examples using SDK for Python \(Boto3\) \(p. 3943\)](#)
- [IAM examples using SDK for Python \(Boto3\) \(p. 3963\)](#)
- [AWS KMS examples using SDK for Python \(Boto3\) \(p. 4006\)](#)
- [Amazon Keyspaces examples using SDK for Python \(Boto3\) \(p. 4025\)](#)
- [Kinesis examples using SDK for Python \(Boto3\) \(p. 4043\)](#)
- [Kinesis Data Analytics examples using SDK for Python \(Boto3\) \(p. 4047\)](#)
- [Lambda examples using SDK for Python \(Boto3\) \(p. 4063\)](#)
- [Lookout for Vision examples using SDK for Python \(Boto3\) \(p. 4076\)](#)
- [Organizations examples using SDK for Python \(Boto3\) \(p. 4095\)](#)
- [Amazon Pinpoint examples using SDK for Python \(Boto3\) \(p. 4099\)](#)
- [Amazon Pinpoint SMS and Voice API examples using SDK for Python \(Boto3\) \(p. 4104\)](#)
- [Amazon Polly examples using SDK for Python \(Boto3\) \(p. 4106\)](#)
- [Amazon RDS examples using SDK for Python \(Boto3\) \(p. 4112\)](#)
- [Amazon Rekognition examples using SDK for Python \(Boto3\) \(p. 4132\)](#)
- [Amazon S3 examples using SDK for Python \(Boto3\) \(p. 4163\)](#)
- [S3 Glacier examples using SDK for Python \(Boto3\) \(p. 4201\)](#)
- [Amazon SES examples using SDK for Python \(Boto3\) \(p. 4217\)](#)
- [Amazon SNS examples using SDK for Python \(Boto3\) \(p. 4247\)](#)
- [Amazon SQS examples using SDK for Python \(Boto3\) \(p. 4254\)](#)
- [AWS STS examples using SDK for Python \(Boto3\) \(p. 4263\)](#)
- [Secrets Manager examples using SDK for Python \(Boto3\) \(p. 4274\)](#)
- [Step Functions examples using SDK for Python \(Boto3\) \(p. 4281\)](#)
- [Amazon Textract examples using SDK for Python \(Boto3\) \(p. 4287\)](#)
- [Amazon Transcribe examples using SDK for Python \(Boto3\) \(p. 4292\)](#)

## ACM examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with AWS Certificate Manager.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3738\)](#)
- [Scenarios \(p. 3745\)](#)

## Actions

### Add tags to a certificate

The following code example shows how to add tags to ACM certificates.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
        Encapsulates ACM functions.  
    """  
    def __init__(self, acm_client):  
        """  
            :param acm_client: A Boto3 ACM client.  
        """  
        self.acm_client = acm_client  
  
    def add_tags(self, certificate_arn, tags):  
        """  
            Adds tags to a certificate. Tags are key-value pairs that contain custom  
            metadata.  
            :param certificate_arn: The ARN of the certificate.  
            :param tags: A dictionary of key-value tags to add to the certificate.  
        """  
        try:  
            self.acm_client.add_tags_to_certificate(  
                CertificateArn=certificate_arn,  
                Tags=[{'Key': key, 'Value': value} for key, value in tags.items()])  
            logger.info("Added %s tags to certificate %s.", len(tags), certificate_arn)  
        except ClientError:  
            logger.exception("Couldn't add tags to certificate %s.", certificate_arn)  
        raise
```

- For API details, see [AddTagsToCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

### Delete a certificate

The following code example shows how to delete ACM certificates.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
    Encapsulates ACM functions.  
    """  
    def __init__(self, acm_client):  
        """  
        :param acm_client: A Boto3 ACM client.  
        """  
        self.acm_client = acm_client  
  
    def remove(self, certificate_arn):  
        """  
        Removes a certificate.  
  
        :param certificate_arn: The ARN of the certificate to remove.  
        """  
        try:  
            self.acm_client.delete_certificate(CertificateArn=certificate_arn)  
            logger.info("Removed certificate %s.", certificate_arn)  
        except ClientError:  
            logger.exception("Couldn't remove certificate %s.", certificate_arn)  
            raise
```

- For API details, see [DeleteCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a certificate

The following code example shows how to describe ACM certificates.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
    Encapsulates ACM functions.  
    """  
    def __init__(self, acm_client):  
        """  
        :param acm_client: A Boto3 ACM client.  
        """  
        self.acm_client = acm_client  
  
    def describe(self, certificate_arn):  
        """  
        Gets certificate metadata.  
  
        :param certificate_arn: The Amazon Resource Name (ARN) of the certificate.  
        :return: Metadata about the certificate.  
        """  
        try:  
            response = self.acm_client.describe_certificate(  
                CertificateArn=certificate_arn)  
            certificate = response['Certificate']  
            logger.info(  
                "Got metadata for certificate for domain %s.",
```

```
        certificate['DomainName'])
    except ClientError:
        logger.exception("Couldn't get data for certificate %s.", certificate_arn)
        raise
    else:
        return certificate
```

- For API details, see [DescribeCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a certificate

The following code example shows how to get ACM certificates.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """
    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client

    def get(self, certificate_arn):
        """
        Gets the body and certificate chain of a certificate.

        :param certificate_arn: The ARN of the certificate.
        :return: The body and chain of a certificate.
        """
        try:
            response = self.acm_client.get_certificate(CertificateArn=certificate_arn)
            logger.info("Got certificate %s and its chain.", certificate_arn)
        except ClientError:
            logger.exception("Couldn't get certificate %s.", certificate_arn)
            raise
        else:
            return response
```

- For API details, see [GetCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Import certificates

The following code example shows how to import ACM certificates.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
    Encapsulates ACM functions.  
    """  
    def __init__(self, acm_client):  
        """  
        :param acm_client: A Boto3 ACM client.  
        """  
        self.acm_client = acm_client  
  
    def import_certificate(self, certificate_body, private_key):  
        """  
        Imports a self-signed certificate to ACM.  
  
        :param certificate_body: The body of the certificate, in PEM format.  
        :param private_key: The unencrypted private key of the certificate, in PEM  
                           format.  
        :return: The ARN of the imported certificate.  
        """  
        try:  
            response = self.acm_client.import_certificate(  
                Certificate=certificate_body, PrivateKey=private_key)  
            certificate_arn = response['CertificateArn']  
            logger.info("Imported certificate.")  
        except ClientError:  
            logger.exception("Couldn't import certificate.")  
            raise  
        else:  
            return certificate_arn
```

- For API details, see [ImportCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

## List certificates

The following code example shows how to list ACM certificates.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
    Encapsulates ACM functions.  
    """  
    def __init__(self, acm_client):  
        """  
        :param acm_client: A Boto3 ACM client.  
        """  
        self.acm_client = acm_client  
  
    def list(  
        self, max_items, statuses=None, key_usage=None, extended_key_usage=None,  
        key_types=None):  
        """  
        Lists the certificates for the current account.  
  
        :param max_items: The maximum number of certificates to list.  
        """
```

```
:param statuses: Filters the results to the specified statuses. If None, all
    certificates are included.
:param key_usage: Filters the results to the specified key usages. If None,
    all key usages are included.
:param extended_key_usage: Filters the results to the specified extended key
    usages. If None, all extended key usages are
    included.
:param key_types: Filters the results to the specified key types. If None, all
    key types are included.
:return: The list of certificates.
"""
try:
    kwargs = {'MaxItems': max_items}
    if statuses is not None:
        kwargs['CertificateStatuses'] = statuses
    includes = {}
    if key_usage is not None:
        includes['keyUsage'] = key_usage
    if extended_key_usage is not None:
        includes['extendedKeyUsage'] = extended_key_usage
    if key_types is not None:
        includes['keyTypes'] = key_types
    if includes:
        kwargs['Includes'] = includes
    response = self.acm_client.list_certificates(**kwargs)
    certificates = response['CertificateSummaryList']
    logger.info("Got %s certificates.", len(certificates))
except ClientError:
    logger.exception("Couldn't get certificates.")
    raise
else:
    return certificates
```

- For API details, see [ListCertificates](#) in *AWS SDK for Python (Boto3) API Reference*.

## List tags for a certificate

The following code example shows how to list tags for ACM certificates.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:
    """
    Encapsulates ACM functions.
    """
    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client

    def list_tags(self, certificate_arn):
        """
        Lists the tags attached to a certificate.

        :param certificate_arn: The ARN of the certificate.
        """
```

```
:return: The dictionary of certificate tags.  
"""  
try:  
    response = self.acm_client.list_tags_for_certificate(  
        CertificateArn=certificate_arn)  
    tags = {tag['Key']: tag['Value'] for tag in response['Tags']}    logger.info("Got %s tags for certificates %s.", len(tags), certificate_arn)  
except ClientError:  
    logger.exception("Couldn't get tags for certificate %s.", certificate_arn)  
    raise  
else:  
    return tags
```

- For API details, see [ListTagsForCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

### Remove tags from a certificate

The following code example shows how to remove tags from ACM certificates.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
    Encapsulates ACM functions.  
    """  
    def __init__(self, acm_client):  
        """  
        :param acm_client: A Boto3 ACM client.  
        """  
        self.acm_client = acm_client  
  
    def remove_tags(self, certificate_arn, tags):  
        """  
        Removes tags from a certificate. If the value of a tag is specified, the tag is removed only when the value matches the value of the certificate's tag. Otherwise, the tag is removed regardless of its value.  
  
        :param certificate_arn: The ARN of the certificate.  
        :param tags: The dictionary of tags to remove.  
        """  
        try:  
            cert_tags = []  
            for key, value in tags.items():  
                tag = {'Key': key}  
                if value is not None:  
                    tag['Value'] = value  
                cert_tags.append(tag)  
            self.acm_client.remove_tags_from_certificate(  
                CertificateArn=certificate_arn, Tags=cert_tags)  
            logger.info(  
                "Removed %s tags from certificate %s.", len(tags), certificate_arn)  
        except ClientError:  
            logger.exception(  
                "Couldn't remove tags from certificate %s.", certificate_arn)  
            raise
```

- For API details, see [RemoveTagsFromCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Request validation of a certificate

The following code example shows how to request validation of ACM certificates.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
        Encapsulates ACM functions.  
    """  
  
    def __init__(self, acm_client):  
        """  
            :param acm_client: A Boto3 ACM client.  
        """  
        self.acm_client = acm_client  
  
  
    def request_validation(  
        self, domain, alternate_domains, method, validation_domains=None):  
        """  
            Starts a validation request that results in a new certificate being issued  
            by ACM. DNS validation requires that you add CNAME records to your DNS  
            provider. Email validation sends email to a list of email addresses that  
            are associated with the domain.  
  
            For more information, see Issuing and managing certificates in the ACM  
            user guide.  
            https://docs.aws.amazon.com/acm/latest/userguide/gs.html  
  
            :param domain: The primary domain to associate with the certificate.  
            :param alternate_domains: Subject Alternate Names (SANs) for the certificate.  
            :param method: The validation method, either DNS or EMAIL.  
            :param validation_domains: Alternate domains to use for email validation, when  
                the email domain differs from the primary domain of  
                the certificate.  
            :return: The ARN of the requested certificate.  
        """  
  
        try:  
            kwargs = {  
                'DomainName': domain,  
                'ValidationMethod': method,  
                'SubjectAlternativeNames': alternate_domains}  
            if validation_domains is not None:  
                kwargs['DomainValidationOptions'] = [{  
                    'DomainName': key,  
                    'ValidationDomain': value  
                } for key, value in validation_domains.items()]  
            response = self.acm_client.request_certificate(**kwargs)  
            certificate_arn = response['CertificateArn']  
            logger.info(  
                "Requested %s validation for domain %s. Certificate ARN is %s.",  
                method, domain, certificate_arn)  
        except ClientError:  
            logger.exception(  
                "Request for %s validation of domain %s failed.", method, domain)  
            raise  
        else:  
            pass
```

```
    return certificate_arn
```

- For API details, see [RequestCertificate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Resend validation email for a certificate

The following code example shows how to resend validation email for ACM certificates.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AcmCertificate:  
    """  
    Encapsulates ACM functions.  
    """  
    def __init__(self, acm_client):  
        """  
        :param acm_client: A Boto3 ACM client.  
        """  
        self.acm_client = acm_client  
  
    def resend_validation_email(self, certificate_arn, domain, validation_domain):  
        """  
        Request that validation email is sent again, for a certificate that was  
        previously requested with email validation.  
  
        :param certificate_arn: The ARN of the certificate.  
        :param domain: The primary domain of the certificate.  
        :param validation_domain: Alternate domain to use for determining email  
            addresses to use for validation.  
        """  
        try:  
            self.acm_client.resend_validation_email(  
                CertificateArn=certificate_arn,  
                Domain=domain,  
                ValidationDomain=validation_domain)  
            logger.info(  
                "Validation email resent to validation domain %s.", validation_domain)  
        except ClientError:  
            logger.exception(  
                "Couldn't resend validation email to %s.", validation_domain)  
            raise
```

- For API details, see [ResendValidationEmail](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Manage certificates

The following code example shows how to:

- Request a certificate from ACM.
- Import a self-signed certificate.

- List and describe certificates.
- Remove certificates.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps ACM operations.

```
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class AcmCertificate:
    """
    Encapsulates ACM functions.
    """
    def __init__(self, acm_client):
        """
        :param acm_client: A Boto3 ACM client.
        """
        self.acm_client = acm_client

    def request_validation(
            self, domain, alternate_domains, method, validation_domains=None):
        """
        Starts a validation request that results in a new certificate being issued
        by ACM. DNS validation requires that you add CNAME records to your DNS
        provider. Email validation sends email to a list of email addresses that
        are associated with the domain.

        For more information, see _Issuing and managing certificates_ in the ACM
        user guide.
        https://docs.aws.amazon.com/acm/latest/userguide/gs.html

        :param domain: The primary domain to associate with the certificate.
        :param alternate_domains: Subject Alternative Names (SANs) for the certificate.
        :param method: The validation method, either DNS or EMAIL.
        :param validation_domains: Alternate domains to use for email validation, when
                                   the email domain differs from the primary domain of
                                   the certificate.
        :return: The ARN of the requested certificate.
        """
        try:
            kwargs = {
                'DomainName': domain,
                'ValidationMethod': method,
                'SubjectAlternativeNames': alternate_domains}
            if validation_domains is not None:
                kwargs['DomainValidationOptions'] = [
                    {
                        'DomainName': key,
                        'ValidationDomain': value
                    } for key, value in validation_domains.items()]
            response = self.acm_client.request_certificate(**kwargs)
            certificate_arn = response['CertificateArn']
            logger.info(
```

```
        "Requested %s validation for domain %s. Certificate ARN is %s.",  
        method, domain, certificate_arn)  
    except ClientError:  
        logger.exception(  
            "Request for %s validation of domain %s failed.", method, domain)  
        raise  
    else:  
        return certificate_arn  
  
def import_certificate(self, certificate_body, private_key):  
    """  
    Imports a self-signed certificate to ACM.  
  
    :param certificate_body: The body of the certificate, in PEM format.  
    :param private_key: The unencrypted private key of the certificate, in PEM  
        format.  
    :return: The ARN of the imported certificate.  
    """  
    try:  
        response = self.acm_client.import_certificate(  
            Certificate=certificate_body, PrivateKey=private_key)  
        certificate_arn = response['CertificateArn']  
        logger.info("Imported certificate.")  
    except ClientError:  
        logger.exception("Couldn't import certificate.")  
        raise  
    else:  
        return certificate_arn  
  
def list(  
    self, max_items, statuses=None, key_usage=None, extended_key_usage=None,  
    key_types=None):  
    """  
    Lists the certificates for the current account.  
  
    :param max_items: The maximum number of certificates to list.  
    :param statuses: Filters the results to the specified statuses. If None, all  
        certificates are included.  
    :param key_usage: Filters the results to the specified key usages. If None,  
        all key usages are included.  
    :param extended_key_usage: Filters the results to the specified extended key  
        usages. If None, all extended key usages are  
        included.  
    :param key_types: Filters the results to the specified key types. If None, all  
        key types are included.  
    :return: The list of certificates.  
    """  
    try:  
        kwargs = {'MaxItems': max_items}  
        if statuses is not None:  
            kwargs['CertificateStatuses'] = statuses  
        includes = {}  
        if key_usage is not None:  
            includes['keyUsage'] = key_usage  
        if extended_key_usage is not None:  
            includes['extendedKeyUsage'] = extended_key_usage  
        if key_types is not None:  
            includes['keyTypes'] = key_types  
        if includes:  
            kwargs['Includes'] = includes  
        response = self.acm_client.list_certificates(**kwargs)  
        certificates = response['CertificateSummaryList']  
        logger.info("Got %s certificates.", len(certificates))  
    except ClientError:  
        logger.exception("Couldn't get certificates.")  
        raise
```

```
        else:
            return certificates

    def describe(self, certificate_arn):
        """
        Gets certificate metadata.

        :param certificate_arn: The Amazon Resource Name (ARN) of the certificate.
        :return: Metadata about the certificate.
        """
        try:
            response = self.acm_client.describe_certificate(
                CertificateArn=certificate_arn)
            certificate = response['Certificate']
            logger.info(
                "Got metadata for certificate for domain %s.",
                certificate['DomainName'])
        except ClientError:
            logger.exception("Couldn't get data for certificate %s.", certificate_arn)
            raise
        else:
            return certificate

    def get(self, certificate_arn):
        """
        Gets the body and certificate chain of a certificate.

        :param certificate_arn: The ARN of the certificate.
        :return: The body and chain of a certificate.
        """
        try:
            response = self.acm_client.get_certificate(CertificateArn=certificate_arn)
            logger.info("Got certificate %s and its chain.", certificate_arn)
        except ClientError:
            logger.exception("Couldn't get certificate %s.", certificate_arn)
            raise
        else:
            return response

    def add_tags(self, certificate_arn, tags):
        """
        Adds tags to a certificate. Tags are key-value pairs that contain custom
        metadata.

        :param certificate_arn: The ARN of the certificate.
        :param tags: A dictionary of key-value tags to add to the certificate.
        """
        try:
            self.acm_client.add_tags_to_certificate(
                CertificateArn=certificate_arn,
                Tags=[{'Key': key, 'Value': value} for key, value in tags.items()])
            logger.info("Added %s tags to certificate %s.", len(tags), certificate_arn)
        except ClientError:
            logger.exception("Couldn't add tags to certificate %s.", certificate_arn)
            raise

    def list_tags(self, certificate_arn):
        """
        Lists the tags attached to a certificate.

        :param certificate_arn: The ARN of the certificate.
        :return: The dictionary of certificate tags.
        """
        try:
            response = self.acm_client.list_tags_for_certificate(
                CertificateArn=certificate_arn)
```

```

tags = {tag['Key']: tag['Value'] for tag in response['Tags']}
logger.info("Got %s tags for certificates %s.", len(tags), certificate_arn)
except ClientError:
    logger.exception("Couldn't get tags for certificate %s.", certificate_arn)
    raise
else:
    return tags

def remove_tags(self, certificate_arn, tags):
    """
    Removes tags from a certificate. If the value of a tag is specified, the tag is
    removed only when the value matches the value of the certificate's tag.
    Otherwise, the tag is removed regardless of its value.

    :param certificate_arn: The ARN of the certificate.
    :param tags: The dictionary of tags to remove.
    """
    try:
        cert_tags = []
        for key, value in tags.items():
            tag = {'Key': key}
            if value is not None:
                tag['Value'] = value
            cert_tags.append(tag)
        self.acm_client.remove_tags_from_certificate(
            CertificateArn=certificate_arn, Tags=cert_tags)
        logger.info(
            "Removed %s tags from certificate %s.", len(tags), certificate_arn)
    except ClientError:
        logger.exception(
            "Couldn't remove tags from certificate %s.", certificate_arn)
        raise

def remove(self, certificate_arn):
    """
    Removes a certificate.

    :param certificate_arn: The ARN of the certificate to remove.
    """
    try:
        self.acm_client.delete_certificate(CertificateArn=certificate_arn)
        logger.info("Removed certificate %s.", certificate_arn)
    except ClientError:
        logger.exception("Couldn't remove certificate %s.", certificate_arn)
        raise

```

Use the wrapper class to manage certificates for your account.

```

def usage_demo():
    print('*'*88)
    print("Welcome to the AWS Certificate Manager (ACM) demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    acm_certificate = AcmCertificate(boto3.client('acm'))
    domain = 'example.com'
    sub_domains = [f'{sub}.{domain}' for sub in ['test', 'dev']]
    print(f'Request a certificate for {domain}.')
    certificate_arn = acm_certificate.request_validation(domain, sub_domains, 'DNS')
    print(f'Started validation, got certificate ARN: {certificate_arn}.')

    import_cert_arn = None
    cert_file_name = input(

```

```
"Enter the file name for a self-signed certificate in PEM format. "
"This certificate will be imported to ACM. Press Enter to skip: ")
if cert_file_name:
    pk_file_name = input(
        "Enter the file name for the unencrypted private key of the certificate. "
        "This file must also be in PEM format: ")
    if pk_file_name:
        with open(cert_file_name, 'rb') as cert_file:
            import_cert = cert_file.read()
        with open(pk_file_name, 'rb') as pk_file:
            import_pk = pk_file.read()
        import_cert_arn = acm_certificate.import_certificate(import_cert,
import_pk)
        print(f"Certificate imported, got ARN: {import_cert_arn}")
    else:
        print("No private key file entered. Skipping certificate import.")
else:
    print("Skipping self-signed certificate import.")

print("Getting the first 10 issued certificates.")
certificates = acm_certificate.list(10, statuses=['ISSUED'])
print(f"Found {len(certificates)} issued certificates.")

print(f"Getting metadata for certificate {certificate_arn}")
cert_metadata = acm_certificate.describe(certificate_arn)
pprint(cert_metadata)

if import_cert_arn is not None:
    print(f"Getting certificate for imported certificate {import_cert_arn}")
    import_cert_data = acm_certificate.get(import_cert_arn)
    pprint(import_cert_data)

print(f"Adding tags to certificate {certificate_arn}.")
acm_certificate.add_tags(certificate_arn, {
    'purpose': 'acm demo',
    'color': 'green'})
tags = acm_certificate.list_tags(certificate_arn)
print(f"Found tags: {tags}")
acm_certificate.remove_tags(certificate_arn, {key: None for key in tags})
print("Removed tags.")

print("Removing certificates added during the demo.")
acm_certificate.remove(certificate_arn)
if import_cert_arn is not None:
    acm_certificate.remove(import_cert_arn)

print("Thanks for watching!")
print('*'*88)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [AddTagsToCertificate](#)
  - [DeleteCertificate](#)
  - [DescribeCertificate](#)
  - [GetCertificate](#)
  - [ImportCertificate](#)
  - [ListCertificates](#)
  - [ListTagsForCertificate](#)
  - [RemoveTagsFromCertificate](#)
  - [RequestCertificate](#)
  - [ResendValidationEmail](#)

## API Gateway examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon API Gateway.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3751\)](#)
- [Scenarios \(p. 3762\)](#)

## Actions

### Add a response from an AWS service integrated with a method

The following code example shows how to add a response from an AWS service integrated with an API Gateway method.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:  
    """  
        Encapsulates Amazon API Gateway functions that are used to create a REST API that  
        integrates with another AWS service.  
    """  
    def __init__(self, apig_client):  
        """  
            :param apig_client: A Boto3 API Gateway client.  
        """  
        self.apig_client = apig_client  
        self.api_id = None  
        self.root_id = None  
        self.stage = None  
  
    def add_integration_method(  
        self, resource_id, rest_method, service_endpoint_prefix, service_action,  
        service_method, role_arn, mapping_template):  
        """  
            Adds an integration method to a REST API. An integration method is a REST  
            resource, such as '/users', and an HTTP verb, such as GET. The integration  
            method is backed by an AWS service, such as Amazon DynamoDB.  
  
            :param resource_id: The ID of the REST resource.  
            :param rest_method: The HTTP verb used with the REST resource.  
            :param service_endpoint_prefix: The service endpoint that is integrated with  
                this method, such as 'dynamodb'.  
            :param service_action: The action that is called on the service, such as  
                'GetItem'.  
            :param service_method: The HTTP method of the service request, such as POST.  
            :param role_arn: The Amazon Resource Name (ARN) of a role that grants API  
                Gateway permission to use the specified action with the  
                service.  
            :param mapping_template: A mapping template that is used to translate REST  
        """
```

```
elements, such as query parameters, to the request
body format required by the service.

"""
service_uri = f'arn:aws:apigateway:{self.apig_client.meta.region_name}'
               f':{service_endpoint_prefix}:action/{service_action}')
try:
    self.apig_client.put_method(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        authorizationType='NONE')
    self.apig_client.put_method_response(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        statusCode='200',
        responseModels={'application/json': 'Empty'})
    logger.info("Created %s method for resource %s.", rest_method, resource_id)
except ClientError:
    logger.exception(
        "Couldn't create %s method for resource %s.", rest_method, resource_id)
    raise

try:
    self.apig_client.put_integration(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        type='AWS',
        integrationHttpMethod=service_method,
        credentials=role_arn,
        requestTemplates={'application/json': json.dumps(mapping_template)},
        uri=service_uri,
        passthroughBehavior='WHEN_NO_TEMPLATES')
    self.apig_client.put_integration_response(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        statusCode='200',
        responseTemplates={'application/json': ''})
    logger.info(
        "Created integration for resource %s to service URI %s.", resource_id,
        service_uri)
except ClientError:
    logger.exception(
        "Couldn't create integration for resource %s to service URI %s.",
        resource_id, service_uri)
    raise
```

- For API details, see [PutIntegrationResponse](#) in *AWS SDK for Python (Boto3) API Reference*.

## Add an HTTP method to a REST resource

The following code example shows how to add an HTTP method to an API Gateway REST resource.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:
```

```

"""
Encapsulates Amazon API Gateway functions that are used to create a REST API that
integrates with another AWS service.
"""
def __init__(self, apig_client):
    """
    :param apig_client: A Boto3 API Gateway client.
    """
    self.apig_client = apig_client
    self.api_id = None
    self.root_id = None
    self.stage = None

    def add_integration_method(
        self, resource_id, rest_method, service_endpoint_prefix, service_action,
        service_method, role_arn, mapping_template):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated with
            this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
            'GetItem'.
        :param service_method: The HTTP method of the service request, such as POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
            Gateway permission to use the specified action with the
            service.
        :param mapping_template: A mapping template that is used to translate REST
            elements, such as query parameters, to the request
            body format required by the service.
        """
        service_uri = (f'arn:aws:apigateway:{self.apig_client.meta.region_name}'
                      f':{service_endpoint_prefix}:action/{service_action}')
        try:
            self.apig_client.put_method(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                authorizationType='NONE')
            self.apig_client.put_method_response(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                statusCode='200',
                responseModels={'application/json': 'Empty'})
            logger.info("Created %s method for resource %s.", rest_method, resource_id)
        except ClientError:
            logger.exception(
                "Couldn't create %s method for resource %s.", rest_method, resource_id)
            raise

        try:
            self.apig_client.put_integration(
                restApiId=self.api_id,
                resourceId=resource_id,
                httpMethod=rest_method,
                type='AWS',
                integrationHttpMethod=service_method,
                credentials=role_arn,
                requestTemplates={'application/json': json.dumps(mapping_template)},
                uri=service_uri,
                passthroughBehavior='WHEN_NO_TEMPLATES')

```

```
        self.apig_client.put_integration_response(
            restApiId=self.api_id,
            resourceId=resource_id,
            httpMethod=rest_method,
            statusCode='200',
            responseTemplates={'application/json': ''})
        logger.info(
            "Created integration for resource %s to service URI %s.", resource_id,
            service_uri)
    except ClientError:
        logger.exception(
            "Couldn't create integration for resource %s to service URI %s.", resource_id, service_uri)
        raise
```

- For API details, see [PutMethod](#) in *AWS SDK for Python (Boto3) API Reference*.

## Add an HTTP response to a REST resource

The following code example shows how to add an HTTP response to an API Gateway REST resource.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API that
    integrates with another AWS service.
    """
    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None

    def add_integration_method(
            self, resource_id, rest_method, service_endpoint_prefix, service_action,
            service_method, role_arn, mapping_template):
        """
        Adds an integration method to a REST API. An integration method is a REST
        resource, such as '/users', and an HTTP verb, such as GET. The integration
        method is backed by an AWS service, such as Amazon DynamoDB.

        :param resource_id: The ID of the REST resource.
        :param rest_method: The HTTP verb used with the REST resource.
        :param service_endpoint_prefix: The service endpoint that is integrated with
                                         this method, such as 'dynamodb'.
        :param service_action: The action that is called on the service, such as
                              'GetItem'.
        :param service_method: The HTTP method of the service request, such as POST.
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                        Gateway permission to use the specified action with the
                        service.
        :param mapping_template: A mapping template that is used to translate REST
                                elements, such as query parameters, to the request
```

```
        body format required by the service.  
"""  
    service_uri = (f'arn:aws:apigateway:{self.apig_client.meta.region_name}'  
                  f':{service_endpoint_prefix}:action/{service_action}')  
    try:  
        self.apig_client.put_method(  
            restApiId=self.api_id,  
            resourceId=resource_id,  
            httpMethod=rest_method,  
            authorizationType='NONE')  
        self.apig_client.put_method_response(  
            restApiId=self.api_id,  
            resourceId=resource_id,  
            httpMethod=rest_method,  
            statusCode='200',  
            responseModels={'application/json': 'Empty'})  
        logger.info("Created %s method for resource %s.", rest_method, resource_id)  
    except ClientError:  
        logger.exception(  
            "Couldn't create %s method for resource %s.", rest_method, resource_id)  
        raise  
  
    try:  
        self.apig_client.put_integration(  
            restApiId=self.api_id,  
            resourceId=resource_id,  
            httpMethod=rest_method,  
            type='AWS',  
            integrationHttpMethod=service_method,  
            credentials=role_arn,  
            requestTemplates={'application/json': json.dumps(mapping_template)},  
            uri=service_uri,  
            passthroughBehavior='WHEN_NO_TEMPLATES')  
        self.apig_client.put_integration_response(  
            restApiId=self.api_id,  
            resourceId=resource_id,  
            httpMethod=rest_method,  
            statusCode='200',  
            responseTemplates={'application/json': ''})  
        logger.info(  
            "Created integration for resource %s to service URI %s.", resource_id,  
            service_uri)  
    except ClientError:  
        logger.exception(  
            "Couldn't create integration for resource %s to service URI %s.",  
            resource_id, service_uri)  
        raise
```

- For API details, see [PutMethodResponse](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a REST API

The following code example shows how to create an API Gateway REST API.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:
```

```
"""
Encapsulates Amazon API Gateway functions that are used to create a REST API that
integrates with another AWS service.
"""
def __init__(self, apig_client):
    """
    :param apig_client: A Boto3 API Gateway client.
    """
    self.apig_client = apig_client
    self.api_id = None
    self.root_id = None
    self.stage = None

def create_rest_api(self, api_name):
    """
    Creates a REST API on API Gateway. The default API has only a root resource
    and no HTTP methods.

    :param api_name: The name of the API. This descriptive name is not used in
                     the API path.
    :return: The ID of the newly created API.
    """
    try:
        result = self.apig_client.create_rest_api(name=api_name)
        self.api_id = result['id']
        logger.info("Created REST API %s with ID %s.", api_name, self.api_id)
    except ClientError:
        logger.exception("Couldn't create REST API %s.", api_name)
        raise

    try:
        result = self.apig_client.get_resources(restApiId=self.api_id)
        self.root_id = next(
            item for item in result['items'] if item['path'] == '/')['id']
    except ClientError:
        logger.exception("Couldn't get resources for API %s.", self.api_id)
        raise
    except StopIteration as err:
        logger.exception("No root resource found in API %s.", self.api_id)
        raise ValueError from err

    return self.api_id
```

- For API details, see [CreateRestApi](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a REST resource

The following code example shows how to create an API Gateway REST resource.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API that
    integrates with another AWS service.
    """
    def __init__(self, apig_client):
        """
```

```
:param apig_client: A Boto3 API Gateway client.  
"""  
    self.apig_client = apig_client  
    self.api_id = None  
    self.root_id = None  
    self.stage = None  
  
def add_rest_resource(self, parent_id, resource_path):  
    """  
        Adds a resource to a REST API.  
  
    :param parent_id: The ID of the parent resource.  
    :param resource_path: The path of the new resource, relative to the parent.  
    :return: The ID of the new resource.  
    """  
    try:  
        result = self.apig_client.create_resource(  
            restApiId=self.api_id, parentId=parent_id, pathPart=resource_path)  
        resource_id = result['id']  
        logger.info("Created resource %s.", resource_path)  
    except ClientError:  
        logger.exception("Couldn't create resource %s.", resource_path)  
        raise  
    else:  
        return resource_id
```

- For API details, see [CreateResource in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete a REST API

The following code example shows how to delete an API Gateway REST API.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:  
    """  
        Encapsulates Amazon API Gateway functions that are used to create a REST API that  
        integrates with another AWS service.  
    """  
    def __init__(self, apig_client):  
        """  
            :param apig_client: A Boto3 API Gateway client.  
        """  
        self.apig_client = apig_client  
        self.api_id = None  
        self.root_id = None  
        self.stage = None  
  
    def delete_rest_api(self):  
        """  
            Deletes a REST API, including all of its resources and configuration.  
        """  
        try:  
            self.apig_client.delete_rest_api(restApiId=self.api_id)  
            logger.info("Deleted REST API %s.", self.api_id)  
            self.api_id = None  
        except ClientError:  
            logger.exception("Couldn't delete REST API %s.", self.api_id)
```

```
    raise
```

- For API details, see [DeleteRestApi](#) in *AWS SDK for Python (Boto3) API Reference*.

## Deploy a REST API

The following code example shows how to deploy an API Gateway REST API.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:  
    """  
        Encapsulates Amazon API Gateway functions that are used to create a REST API that  
        integrates with another AWS service.  
    """  
    def __init__(self, apig_client):  
        """  
            :param apig_client: A Boto3 API Gateway client.  
        """  
        self.apig_client = apig_client  
        self.api_id = None  
        self.root_id = None  
        self.stage = None  
  
    def deploy_api(self, stage_name):  
        """  
            Deploys a REST API. After a REST API is deployed, it can be called from any  
            REST client, such as the Python Requests package or Postman.  
  
            :param stage_name: The stage of the API to deploy, such as 'test'.  
            :return: The base URL of the deployed REST API.  
        """  
        try:  
            self.apig_client.create_deployment(  
                restApiId=self.api_id, stageName=stage_name)  
            self.stage = stage_name  
            logger.info("Deployed stage %s.", stage_name)  
        except ClientError:  
            logger.exception("Couldn't deploy stage %s.", stage_name)  
            raise  
        else:  
            return self.api_url()  
  
    def api_url(self, resource=None):  
        """  
            Builds the REST API URL from its parts.  
  
            :param resource: The resource path to append to the base URL.  
            :return: The REST URL to the specified resource.  
        """  
        url = (f'https://{{self.api_id}}.execute-api.{{self.apig_client.meta.region_name}}'  
              f'.amazonaws.com/{{self.stage}}')  
        if resource is not None:  
            url = f'{url}/{resource}'  
        return url
```

- For API details, see [CreateDeployment](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get REST resources

The following code example shows how to get API Gateway REST resources.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:  
    """  
        Encapsulates Amazon API Gateway functions that are used to create a REST API that  
        integrates with another AWS service.  
    """  
    def __init__(self, apig_client):  
        """  
            :param apig_client: A Boto3 API Gateway client.  
        """  
        self.apig_client = apig_client  
        self.api_id = None  
        self.root_id = None  
        self.stage = None  
  
    def create_rest_api(self, api_name):  
        """  
            Creates a REST API on API Gateway. The default API has only a root resource  
            and no HTTP methods.  
  
            :param api_name: The name of the API. This descriptive name is not used in  
                the API path.  
            :return: The ID of the newly created API.  
        """  
        try:  
            result = self.apig_client.create_rest_api(name=api_name)  
            self.api_id = result['id']  
            logger.info("Created REST API %s with ID %s.", api_name, self.api_id)  
        except ClientError:  
            logger.exception("Couldn't create REST API %s.", api_name)  
            raise  
  
        try:  
            result = self.apig_client.get_resources(restApiId=self.api_id)  
            self.root_id = next(  
                item for item in result['items'] if item['path'] == '/')['id']  
        except ClientError:  
            logger.exception("Couldn't get resources for API %s.", self.api_id)  
            raise  
        except StopIteration as err:  
            logger.exception("No root resource found in API %s.", self.api_id)  
            raise ValueError from err  
  
    return self.api_id
```

- For API details, see [GetResources](#) in *AWS SDK for Python (Boto3) API Reference*.

## Integrate a method with an AWS service

The following code example shows how to integrate an API Gateway method with an AWS service.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:  
    """  
    Encapsulates Amazon API Gateway functions that are used to create a REST API that  
    integrates with another AWS service.  
    """  
    def __init__(self, apig_client):  
        """  
        :param apig_client: A Boto3 API Gateway client.  
        """  
        self.apig_client = apig_client  
        self.api_id = None  
        self.root_id = None  
        self.stage = None  
  
    def add_integration_method(  
            self, resource_id, rest_method, service_endpoint_prefix, service_action,  
            service_method, role_arn, mapping_template):  
        """  
        Adds an integration method to a REST API. An integration method is a REST  
        resource, such as '/users', and an HTTP verb, such as GET. The integration  
        method is backed by an AWS service, such as Amazon DynamoDB.  
  
        :param resource_id: The ID of the REST resource.  
        :param rest_method: The HTTP verb used with the REST resource.  
        :param service_endpoint_prefix: The service endpoint that is integrated with  
            this method, such as 'dynamodb'.  
        :param service_action: The action that is called on the service, such as  
            'GetItem'.  
        :param service_method: The HTTP method of the service request, such as POST.  
        :param role_arn: The Amazon Resource Name (ARN) of a role that grants API  
            Gateway permission to use the specified action with the  
            service.  
        :param mapping_template: A mapping template that is used to translate REST  
            elements, such as query parameters, to the request  
            body format required by the service.  
        """  
        service_uri = (f'arn:aws:apigateway:{self.apig_client.meta.region_name}'  
                      f':{service_endpoint_prefix}:action/{service_action}')  
        try:  
            self.apig_client.put_method(  
                restApiId=self.api_id,  
                resourceId=resource_id,  
                httpMethod=rest_method,  
                authorizationType='NONE')  
            self.apig_client.put_method_response(  
                restApiId=self.api_id,  
                resourceId=resource_id,  
                httpMethod=rest_method,  
                statusCode='200',  
                responseModels={'application/json': 'Empty'})  
            logger.info("Created %s method for resource %s.", rest_method, resource_id)  
        except ClientError:  
            logger.exception(  
                "Couldn't create %s method for resource %s.", rest_method, resource_id)  
            raise  
  
        try:  
            self.apig_client.put_integration(  
                restApiId=self.api_id,  
                resourceId=resource_id,  
                httpMethod=rest_method,  
                type='AWS_PROXY',  
                integrationHttpMethod='POST',  
                uri=service_uri,  
                integrationResponses=[  
                    {'statusCode': '200',  
                     'responseParameters': {  
                         'method.response.header.Content-Type': "'application/json'"},  
                     'responseTemplates': {  
                         'application/json': '{'}}]  
            )  
            logger.info("Created integration for resource %s.", resource_id)  
        except ClientError:  
            logger.exception(  
                "Couldn't create integration for resource %s.", resource_id)  
            raise
```

```
restApiId=self.api_id,
resourceId=resource_id,
httpMethod=rest_method,
type='AWS',
integrationHttpMethod=service_method,
credentials=role_arn,
requestTemplates={'application/json': json.dumps(mapping_template)},
uri=service_uri,
passthroughBehavior='WHEN_NO_TEMPLATES')
self.apig_client.put_integration_response(
    restApiId=self.api_id,
    resourceId=resource_id,
    httpMethod=rest_method,
    statusCode='200',
    responseTemplates={'application/json': ''})
logger.info(
    "Created integration for resource %s to service URI %s.", resource_id,
    service_uri)
except ClientError:
    logger.exception(
        "Couldn't create integration for resource %s to service URI %s.",
        resource_id, service_uri)
raise
```

- For API details, see [PutIntegration](#) in *AWS SDK for Python (Boto3) API Reference*.

## List REST APIs

The following code example shows how to list API Gateway REST APIs.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API that
    integrates with another AWS service.
    """
    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None

    def get_rest_api_id(self, api_name):
        """
        Gets the ID of a REST API from its name by searching the list of REST APIs
        for the current account. Because names need not be unique, this returns only
        the first API with the specified name.

        :param api_name: The name of the API to look up.
        :return: The ID of the specified API.
        """
        try:
            rest_api = None
            paginator = self.apig_client.getPaginator('get_rest_apis')
```

```
        for page in paginator.paginate():
            rest_api = next(
                (item for item in page['items'] if item['name'] == api_name), None)
            if rest_api is not None:
                break
            self.api_id = rest_api['id']
            logger.info("Found ID %s for API %s.", rest_api['id'], api_name)
        except ClientError:
            logger.exception("Couldn't find ID for API %s.", api_name)
            raise
        else:
            return rest_api['id']
```

- For API details, see [GetRestApis](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Create and deploy a REST API

The following code example shows how to:

- Create a REST API served by API Gateway.
- Add resources to the REST API to represent a user profile.
- Add integration methods so that the REST API uses a DynamoDB table to store user profile data.
- Send HTTP requests to the REST API to add and retrieve user profiles.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps API Gateway operations.

```
import argparse
import json
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)

class ApiGatewayToService:
    """
    Encapsulates Amazon API Gateway functions that are used to create a REST API that
    integrates with another AWS service.
    """
    def __init__(self, apig_client):
        """
        :param apig_client: A Boto3 API Gateway client.
        """
        self.apig_client = apig_client
        self.api_id = None
        self.root_id = None
        self.stage = None

    def create_rest_api(self, api_name):
```

```

"""
Creates a REST API on API Gateway. The default API has only a root resource
and no HTTP methods.

:param api_name: The name of the API. This descriptive name is not used in
                 the API path.
:return: The ID of the newly created API.
"""
try:
    result = self.apig_client.create_rest_api(name=api_name)
    self.api_id = result['id']
    logger.info("Created REST API %s with ID %s.", api_name, self.api_id)
except ClientError:
    logger.exception("Couldn't create REST API %s.", api_name)
    raise

try:
    result = self.apig_client.get_resources(restApiId=self.api_id)
    self.root_id = next(
        item for item in result['items'] if item['path'] == '/')['id']
except ClientError:
    logger.exception("Couldn't get resources for API %s.", self.api_id)
    raise
except StopIteration as err:
    logger.exception("No root resource found in API %s.", self.api_id)
    raise ValueError from err

return self.api_id

def add_rest_resource(self, parent_id, resource_path):
    """
    Adds a resource to a REST API.

    :param parent_id: The ID of the parent resource.
    :param resource_path: The path of the new resource, relative to the parent.
    :return: The ID of the new resource.
    """
    try:
        result = self.apig_client.create_resource(
            restApiId=self.api_id, parentId=parent_id, pathPart=resource_path)
        resource_id = result['id']
        logger.info("Created resource %s.", resource_path)
    except ClientError:
        logger.exception("Couldn't create resource %s.", resource_path)
        raise
    else:
        return resource_id

def add_integration_method(
    self, resource_id, rest_method, service_endpoint_prefix, service_action,
    service_method, role_arn, mapping_template):
    """
    Adds an integration method to a REST API. An integration method is a REST
    resource, such as '/users', and an HTTP verb, such as GET. The integration
    method is backed by an AWS service, such as Amazon DynamoDB.

    :param resource_id: The ID of the REST resource.
    :param rest_method: The HTTP verb used with the REST resource.
    :param service_endpoint_prefix: The service endpoint that is integrated with
                                   this method, such as 'dynamodb'.
    :param service_action: The action that is called on the service, such as
                          'GetItem'.
    :param service_method: The HTTP method of the service request, such as POST.
    :param role_arn: The Amazon Resource Name (ARN) of a role that grants API
                    Gateway permission to use the specified action with the
                    service.
    """

```

```

:param mapping_template: A mapping template that is used to translate REST
elements, such as query parameters, to the request
body format required by the service.
"""
service_uri = (f'arn:aws:apigateway:{self.apig_client.meta.region_name}'
               f':{service_endpoint_prefix}:action/{service_action}')
try:
    self.apig_client.put_method(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        authorizationType='NONE')
    self.apig_client.put_method_response(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        statusCode='200',
        responseModels={'application/json': 'Empty'})
    logger.info("Created %s method for resource %s.", rest_method, resource_id)
except ClientError:
    logger.exception(
        "Couldn't create %s method for resource %s.", rest_method, resource_id)
    raise

try:
    self.apig_client.put_integration(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        type='AWS',
        integrationHttpMethod=service_method,
        credentials=role_arn,
        requestTemplates={'application/json': json.dumps(mapping_template)},
        uri=service_uri,
        passthroughBehavior='WHEN_NO_TEMPLATES')
    self.apig_client.put_integration_response(
        restApiId=self.api_id,
        resourceId=resource_id,
        httpMethod=rest_method,
        statusCode='200',
        responseTemplates={'application/json': ''})
    logger.info(
        "Created integration for resource %s to service URI %s.", resource_id,
        service_uri)
except ClientError:
    logger.exception(
        "Couldn't create integration for resource %s to service URI %s.",
        resource_id, service_uri)
    raise

def deploy_api(self, stage_name):
    """
    Deploys a REST API. After a REST API is deployed, it can be called from any
    REST client, such as the Python Requests package or Postman.

    :param stage_name: The stage of the API to deploy, such as 'test'.
    :return: The base URL of the deployed REST API.
    """
    try:
        self.apig_client.create_deployment(
            restApiId=self.api_id, stageName=stage_name)
        self.stage = stage_name
        logger.info("Deployed stage %s.", stage_name)
    except ClientError:
        logger.exception("Couldn't deploy stage %s.", stage_name)
        raise

```

```

        else:
            return self.api_url()

    def api_url(self, resource=None):
        """
        Builds the REST API URL from its parts.

        :param resource: The resource path to append to the base URL.
        :return: The REST URL to the specified resource.
        """
        url = (f'https://{{self.api_id}}.execute-api.{{self.apig_client.meta.region_name}}.'
               f'.amazonaws.com/{{self.stage}}')
        if resource is not None:
            url = f'{url}/{resource}'
        return url

```

Deploy a REST API and call it with the Requests package.

```

def usage_demo(table_name, role_name, rest_api_name):
    """
    Demonstrates how to used API Gateway to create and deploy a REST API, and how
    to use the Requests package to call it.

    :param table_name: The name of the demo DynamoDB table.
    :param role_name: The name of the demo role that grants API Gateway permission to
                      call DynamoDB.
    :param rest_api_name: The name of the demo REST API created by the demo.
    """
    gateway = ApiGatewayToService(boto3.client('apigateway'))
    role = boto3.resource('iam').Role(role_name)

    print("Creating REST API in API Gateway.")
    gateway.create_rest_api(rest_api_name)

    print("Adding resources to the REST API.")
    profiles_id = gateway.add_rest_resource(gateway.root_id, 'profiles')
    username_id = gateway.add_rest_resource(profiles_id, '{username}')

    # The DynamoDB service requires that all integration requests use POST.
    print("Adding integration methods to read and write profiles in Amazon DynamoDB.")
    gateway.add_integration_method(
        profiles_id, 'GET', 'dynamodb', 'Scan', 'POST', role.arn,
        {'TableName': table_name})
    gateway.add_integration_method(
        profiles_id, 'POST', 'dynamodb', 'PutItem', 'POST', role.arn, {
            "TableName": table_name,
            "Item": {
                "username": {"S": "$input.path('$username')"}, "name": {"S": "$input.path('$name')"}, "title": {"S": "$input.path('$title')"}}})
    gateway.add_integration_method(
        username_id, 'GET', 'dynamodb', 'GetItem', 'POST', role.arn, {
            "TableName": table_name,
            "Key": {"username": {"S": "$method.request.path.username"}}})

    stage = 'test'
    print(f"Deploying the {stage} stage.")
    gateway.deploy_api(stage)

    profiles_url = gateway.api_url('profiles')
    print(f"Using the Requests package to post some people to the profiles REST API at "
          f"{{profiles_url}}")

```

```
requests.post(profiles_url, json={
    'username': 'will', 'name': 'William Shakespeare', 'title': 'playwright'})
requests.post(profiles_url, json={
    'username': 'ludwig', 'name': 'Ludwig van Beethoven', 'title': 'composer'})
requests.post(profiles_url, json={
    'username': 'jane', 'name': 'Jane Austen', 'title': 'author'})
print("Getting the list of profiles from the REST API.")
profiles = requests.get(profiles_url).json()
pprint(profiles)
print(f"Getting just the profile for username 'jane' (URL: {profiles_url}/jane).")
jane = requests.get(f'{profiles_url}/jane').json()
pprint(jane)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.

- [CreateDeployment](#)
- [CreateResource](#)
- [CreateRestApi](#)
- [DeleteRestApi](#)
- [GetResources](#)
- [GetRestApis](#)
- [PutIntegration](#)
- [PutIntegrationResponse](#)
- [PutMethod](#)
- [PutMethodResponse](#)

## Application Recovery Controller examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Route 53 Application Recovery Controller.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3766\)](#)

## Actions

### Get the state of a routing control

The following code example shows how to get the state of an Application Recovery Controller routing control.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import boto3
```

```
def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the specified
    cluster endpoint URL and AWS Region.

    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
        'route53-recovery-cluster',
        endpoint_url=cluster_endpoint['Endpoint'],
        region_name=cluster_endpoint['Region'])

def get_routing_control_state(routing_control_arn, cluster_endpoints):
    """
    Gets the state of a routing control. Cluster endpoints are tried in
    sequence until the first successful response is received.

    :param routing_control_arn: The ARN of the routing control to look up.
    :param cluster_endpoints: The list of cluster endpoints to query.
    :return: The routing control state response.
    """
    for cluster_endpoint in cluster_endpoints:
        try:
            recovery_client = create_recovery_client(cluster_endpoint)
            response = recovery_client.get_routing_control_state(
                RoutingControlArn=routing_control_arn)
            return response
        except Exception as error:
            print(error)
```

- For API details, see [GetRoutingControlState](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update the state of a routing control

The following code example shows how to update the state of an Application Recovery Controller routing control.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import boto3

def create_recovery_client(cluster_endpoint):
    """
    Creates a Boto3 Route 53 Application Recovery Controller client for the specified
    cluster endpoint URL and AWS Region.

    :param cluster_endpoint: The cluster endpoint URL and Region.
    :return: The Boto3 client.
    """
    return boto3.client(
        'route53-recovery-cluster',
        endpoint_url=cluster_endpoint['Endpoint'],
        region_name=cluster_endpoint['Region'])

def update_routing_control_state()
```

```
    routing_control_arn, cluster_endpoints, routing_control_state):
"""
Updates the state of a routing control. Cluster endpoints are tried in
sequence until the first successful response is received.

:param routing_control_arn: The ARN of the routing control to update the state for.
:param cluster_endpoints: The list of cluster endpoints to try.
:param routing_control_state: The new routing control state.
:return: The routing control update response.
"""
for cluster_endpoint in cluster_endpoints:
    try:
        recovery_client = create_recovery_client(cluster_endpoint)
        response = recovery_client.update_routing_control_state(
            RoutingControlArn=routing_control_arn,
            RoutingControlState=routing_control_state)
        return response
    except Exception as error:
        print(error)
```

- For API details, see [UpdateRoutingControlState](#) in *AWS SDK for Python (Boto3) API Reference*.

## Audit Manager examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with AWS Audit Manager.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Scenarios \(p. 3768\)](#)

## Scenarios

### Create a custom framework from an AWS Config conformance pack

The following code example shows how to:

- Get a list of AWS Config conformance packs.
- Create an Audit Manager custom control for each managed rule in a conformance pack.
- Create an Audit Manager custom framework that contains the controls.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```

class ConformancePack:
    def __init__(self, config_client, auditmanager_client):
        self.config_client = config_client
        self.auditmanager_client = auditmanager_client

    def get_conformance_pack(self):
        """
        Return a selected conformance pack from the list of conformance packs.

        :return: selected conformance pack
        """
        try:
            conformance_packs = self.config_client.describe_conformance_packs()
            print("Number of conformance packs fetched: ",
                  len(conformance_packs.get("ConformancePackDetails")))
            print("Fetched the following conformance packs: ")
            all_cpack_names = [
                cp['ConformancePackName']
                for cp in conformance_packs.get("ConformancePackDetails")]
            for pack in all_cpack_names:
                print(f"\t{pack}")
            cpack_name = input(
                'Provide ConformancePackName that you want to create a custom '
                'framework for: ')
            if cpack_name not in all_cpack_names:
                print(f'{cpack_name} is not in the list of conformance packs!')
                print('Provide a conformance pack name from the available list of '
                      'conformance packs.')
                raise Exception("Invalid conformance pack")
            print('-' * 88)
        except ClientError:
            logger.exception("Couldn't select conformance pack.")
            raise
        else:
            return cpack_name

    def create_custom_controls(self, cpack_name):
        """
        Create custom controls for all managed AWS Config rules in a conformance pack.

        :param cpack_name: The name of the conformance pack to create controls for.
        :return: The list of custom control IDs.
        """
        try:
            rules_in_pack = self.config_client.describe_conformance_pack_compliance(
                ConformancePackName=cpack_name)
            print('Number of rules in the conformance pack: ',
                  len(rules_in_pack.get('ConformancePackRuleComplianceList')))
            for rule in rules_in_pack.get('ConformancePackRuleComplianceList'):
                print(f"\t{rule.get('ConfigRuleName')}")
            print('-' * 88)
            print('Creating a custom control for each rule and a custom framework '
                  'consisting of these rules in Audit Manager.')
            am_controls = []
            for rule in rules_in_pack.get('ConformancePackRuleComplianceList'):
                config_rule = self.config_client.describe_config_rules(
                    ConfigRuleNames=[rule.get('ConfigRuleName')])
                source_id = config_rule.get('ConfigRules')[0].get('Source', {}).get(
                    'SourceIdentifier')
                custom_control = self.auditmanager_client.create_control(
                    name="Config-" + rule.get('ConfigRuleName'),
                    controlMappingSources=[{
                        'sourceName': 'ConfigRule',
                        'sourceSetUpOption': 'System_Controls_Mapping',
                        'sourceType': 'AWS_Config',
                    }])
                am_controls.append(custom_control)
        except ClientError:
            logger.exception("Couldn't create custom controls for the conformance pack.")
            raise
        else:
            return am_controls

```

```

        'sourceKeyword': {
            'keywordInputType': 'SELECT_FROM_LIST',
            'keywordValue': source_id}]}).get('control', {})
        am_controls.append({'id': custom_control.get('id')})
    print('Successfully created a control for each config rule.')
    print('-' * 88)
except ClientError:
    logger.exception("Failed to create custom controls.")
    raise
else:
    return am_controls

def create_custom_framework(self, cpack_name, am_control_ids):
    """
    Create a custom Audit Manager framework from a selected AWS Config conformance
    pack.

    :param cpack_name: The name of the conformance pack to create a framework from.
    :param am_control_ids: The IDs of the custom controls created from the
                           conformance pack.
    """
    try:
        print('Creating custom framework...')
        custom_framework = self.auditmanager_client.create_assessment_framework(
            name='Config-Conformance-pack-' + cpack_name,
            controlSets=[{'name': cpack_name, 'controls': am_control_ids}])
        print(f"Successfully created the custom framework: ",
              f"{custom_framework.get('framework').get('name')}: ",
              f"{custom_framework.get('framework').get('id')}")
        print('-' * 88)
    except ClientError:
        logger.exception("Failed to create custom framework.")
        raise

def run_demo():
    print('-' * 88)
    print("Welcome to the AWS Audit Manager custom framework demo!")
    print('-' * 88)
    print("You can use this sample to select a conformance pack from AWS Config and "
          "use AWS Audit Manager to create a custom control for all the managed "
          "rules under the conformance pack. A custom framework is also created "
          "with these controls.")
    print('-' * 88)
    conf_pack = ConformancePack(boto3.client('config'), boto3.client('auditmanager'))
    cpack_name = conf_pack.get_conformance_pack()
    am_controls = conf_pack.create_custom_controls(cpack_name)
    conf_pack.create_custom_framework(cpack_name, am_controls)

if __name__ == '__main__':
    run_demo()

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateAssessmentFramework](#)
  - [CreateControl](#)

### Create a custom framework that contains Security Hub controls

The following code example shows how to:

- Get a list of all standard controls that have Security Hub as their data source.

- Create an Audit Manager custom framework that contains the controls.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class SecurityHub:
    def __init__(self, auditmanager_client):
        self.auditmanager_client = auditmanager_client

    def get_sechub_controls(self):
        """
        Gets the list of controls that use Security Hub as their data source.

        :return: The list of Security Hub controls.
        """
        print('-'*88)
        next_token = None
        page = 1
        sechub_control_list = []
        while True:
            print("Page [" + str(page) + "]")
            if next_token is None:
                control_list = self.auditmanager_client.list_controls(
                    controlType='Standard',
                    maxResults=100)
            else:
                control_list = self.auditmanager_client.list_controls(
                    controlType='Standard',
                    nextToken=next_token,
                    maxResults=100)
            print('Total controls found:',
len(control_list.get('controlMetadataList')))
            for control in control_list.get('controlMetadataList'):
                control_details = self.auditmanager_client.get_control(
                    controlId=control.get('id')).get('control', {})
                if "AWS Security Hub" in control_details.get('controlSources'):
                    sechub_control_list.append({'id': control_details.get('id')})
            next_token = control_list.get('nextToken')
            if not next_token:
                break
            page += 1
        print('Number of Security Hub controls found: ', len(sechub_control_list))
        return sechub_control_list

    def create_custom_framework(self, am_controls):
        """
        Create a custom framework with a list of controls.

        :param am_controls: The list of controls to include in the framework.
        """
        try:
            print('Creating custom framework...')
            custom_framework = self.auditmanager_client.create_assessment_framework(
```

```
        name='All Security Hub Controls Framework',
        controlSets=[{'name': "Security-Hub", 'controls': am_controls}])
    print(f"Successfully created the custom framework: "
          f"{custom_framework.get('framework').get('name')}: "
          f"{custom_framework.get('framework').get('id')}")
    print('-' * 88)
except ClientError:
    logger.exception("Failed to create custom framework.")
    raise

def run_demo():
    print('-' * 88)
    print("Welcome to the AWS Audit Manager Security Hub demo!")
    print('-' * 88)
    print(" This script creates a custom framework with all Security Hub controls.")
    print('-' * 88)
    sechub = SecurityHub(boto3.client('auditmanager'))
    am_controls = sechub.get_sechub_controls()
    sechub.create_custom_framework(am_controls)

if __name__ == '__main__':
    run_demo()
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateAssessmentFramework](#)
  - [GetControl](#)
  - [ListControls](#)

### Create an assessment report that contains one day of evidence

The following code example shows how to create an Audit Manager assessment report that contains one day of evidence.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import dateutil.parser
import logging
import time
import urllib.request
import uuid
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class AuditReport:
    def __init__(self, auditmanager_client):
        self.auditmanager_client = auditmanager_client

    def get_input(self):
        print('-' * 40)
        try:
```

```
assessment_id = input('Provide assessment id [uuid]: ').lower()
try:
    assessment_uuid = uuid.UUID(assessment_id)
except ValueError:
    logger.error("Assessment Id is not a valid UUID: %s", assessment_id)
    raise
evidence_folder = input('Provide evidence date [yyyy-mm-dd]: ')
try:
    evidence_date = dateutil.parser.parse(evidence_folder).date()
except ValueError:
    logger.error("Invalid date : %s", evidence_folder)
    raise
try:

self.auditmanager_client.get_assessment(assessmentId=str(assessment_uuid))
except ClientError:
    logger.exception("Couldn't get assessment %s.", assessment_uuid)
    raise
except (ValueError, ClientError):
    return None, None
else:
    return assessment_uuid, evidence_date

def clear_staging(self, assessment_uuid, evidence_date):
"""
Find all the evidence in the report and clear it.
"""
next_token = None
page = 1
interested_folder_id_list = []
while True:
    print(f"Page [{page}]")
    if next_token is None:
        folder_list =
self.auditmanager_client.get_evidence_folders_by_assessment(
    assessmentId=str(assessment_uuid),
    maxResults=1000)
    else:
        folder_list =
self.auditmanager_client.get_evidence_folders_by_assessment(
    assessmentId=str(assessment_uuid),
    nextToken=next_token,
    maxResults=1000)
folders = folder_list.get('evidenceFolders')
print(f"Got {len(folders)} folders.")
for folder in folders:
    folder_id = folder.get('id')
    if folder.get('name') == str(evidence_date):
        interested_folder_id_list.append(folder_id)
    if folder.get('assessmentReportSelectionCount') ==
folder.get('totalEvidence'):
        print(
            f"Removing folder from report selection : {folder.get('name')}"
        )
        f"{{folder_id}} {{folder.get('controlId')}}"

self.auditmanager_client.disassociate_assessment_report_evidence_folder(
    assessmentId=str(assessment_uuid),
    evidenceFolderId=folder_id)
elif folder.get('assessmentReportSelectionCount') > 0:
    # Get all evidence in the folder and
    # add selected evidence in the selected_evidence_list.
    evidence_list =
self.auditmanager_client.get_evidence_by_evidence_folder(
    assessmentId=str(assessment_uuid),
    controlSetId=folder_id,
```

```

        evidenceFolderId=folder_id,
        maxResults=1000)
    selected_evidence_list = []
    for evidence in evidence_list.get('evidence'):
        if evidence.get('assessmentReportSelection') == 'Yes':
            selected_evidence_list.append(evidence.get('id'))
    print(f"Removing evidence report selection : {folder.get('name')} "
          f"[{len(selected_evidence_list)}]")

self.auditmanager_client.batch_disassociate_assessment_report_evidence(
    assessmentId=str(assessment_uuid),
    evidenceFolderId=folder_id,
    evidenceIds=selected_evidence_list)
next_token = folder_list.get('nextToken')
if not next_token:
    break
page += 1
return interested_folder_id_list

def add_folder_to_staging(self, assessment_uuid, folder_id_list):
    print(f"Adding folders to report : {folder_id_list}")
    for folder in folder_id_list:
        self.auditmanager_client.associate_assessment_report_evidence_folder(
            assessmentId=str(assessment_uuid),
            evidenceFolderId=folder)

def get_report(self, assessment_uuid):
    report = self.auditmanager_client.create_assessment_report(
        name='ReportViaScript',
        description='testing',
        assessmentId=str(assessment_uuid))
    if self._is_report_generated(report.get('assessmentReport').get('id')):
        report_url = self.auditmanager_client.get_assessment_report_url(
            assessmentReportId=report.get('assessmentReport').get('id'),
            assessmentId=str(assessment_uuid))
        print(report_url.get('preSignedUrl'))
        urllib.request.urlretrieve(
            report_url.get('preSignedUrl').get('link'),
            report_url.get('preSignedUrl').get('hyperlinkName'))
        print(f"Report saved as
{report_url.get('preSignedUrl').get('hyperlinkName')}")
    else:
        print("Report generation did not finish in 15 minutes.")
        print("Failed to download report. Go to the console and manually download "
              "the report.")

def _is_report_generated(self, assessment_report_id):
    max_wait_time = 0
    while max_wait_time < 900:
        print(f"Checking status of the report {assessment_report_id}")
        report_list =
self.auditmanager_client.list_assessment_reports(maxResults=1)
        if (report_list.get('assessmentReports')[0].get('id') ==
assessment_report_id
            and report_list.get('assessmentReports')[0].get('status') ==
'COMPLETE'):
            return True
        print('Sleeping for 5 seconds...')
        time.sleep(5)
        max_wait_time += 5

def run_demo():
    print('-' * 88)
    print("Welcome to the AWS Audit Manager samples demo!")
    print('-' * 88)

```

```
print("This script creates an assessment report for an assessment with all the "
      "evidence collected on the provided date.")
print('-' * 88)

report = AuditReport(boto3.client('auditmanager'))
assessment_uuid, evidence_date = report.get_input()
if assessment_uuid is not None and evidence_date is not None:
    folder_id_list = report.clear_staging(assessment_uuid, evidence_date)
    report.add_folder_to_staging(assessment_uuid, folder_id_list)
    report.get_report(assessment_uuid)

if __name__ == '__main__':
    run_demo()
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [AssociateAssessmentReportEvidenceFolder](#)
  - [BatchDisassociateAssessmentReportEvidence](#)
  - [CreateAssessmentReport](#)
  - [DisassociateAssessmentReportEvidenceFolder](#)
  - [GetAssessment](#)
  - [GetAssessmentReportUrl](#)
  - [GetEvidenceByEvidenceFolder](#)
  - [GetEvidenceFoldersByAssessment](#)
  - [ListAssessmentReports](#)

## Aurora examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Aurora.

**Actions** are code excerpts that show you how to call individual service functions.

**Scenarios** are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3775\)](#)
- [Scenarios \(p. 3788\)](#)

## Actions

### Create a DB cluster

The following code example shows how to create an Aurora DB cluster.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
```

```
"""
:param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
client.
"""
self.rds_client = rds_client

@classmethod
def from_client(cls):
    """
    Instantiates this class from a Boto3 client.
    """
    rds_client = boto3.client('rds')
    return cls(rds_client)

def create_db_cluster(
        self, cluster_name, parameter_group_name, db_name, db_engine,
        db_engine_version, admin_name, admin_password):
    """
    Creates a DB cluster that is configured to use the specified parameter group.
    The newly created DB cluster contains a database that uses the specified engine
    and
    engine version.

    :param cluster_name: The name of the DB cluster to create.
    :param parameter_group_name: The name of the parameter group to associate with
        the DB cluster.
    :param db_name: The name of the database to create.
    :param db_engine: The database engine of the database that is created, such as
        MySql.
    :param db_engine_version: The version of the database engine.
    :param admin_name: The user name of the database administrator.
    :param admin_password: The password of the database administrator.
    :return: The newly created DB cluster.
    """
try:
    response = self.rds_client.create_db_cluster(
        DatabaseName=db_name,
        DBClusterIdentifier=cluster_name,
        DBClusterParameterGroupName=parameter_group_name,
        Engine=db_engine,
        EngineVersion=db_engine_version,
        MasterUsername=admin_name,
        MasterUserPassword=admin_password)
    cluster = response['DBCluster']
except ClientError as err:
    logger.error(
        "Couldn't create database %s. Here's why: %s: %s",
        db_name,
        err.response['Error']['Code'],
        err.response['Error']['Message'])
    raise
else:
    return cluster
```

- For API details, see [CreateDBCluster in AWS SDK for Python \(Boto3\) API Reference](#).

## Create a DB cluster parameter group

The following code example shows how to create an Aurora DB cluster parameter group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def create_parameter_group(self, parameter_group_name, parameter_group_family,
                               description):
        """
        Creates a DB cluster parameter group that is based on the specified parameter
        group family.

        :param parameter_group_name: The name of the newly created parameter group.
        :param parameter_group_family: The family that is used as the basis of the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description)
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

- For API details, see [CreateDBClusterParameterGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a DB cluster snapshot

The following code example shows how to create an Aurora DB cluster snapshot.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
```

```
:param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
client.
"""
    self.rds_client = rds_client

@classmethod
def from_client(cls):
"""
Instantiates this class from a Boto3 client.
"""
    rds_client = boto3.client('rds')
    return cls(rds_client)

def create_cluster_snapshot(self, snapshot_id, cluster_id):
"""
Creates a snapshot of a DB cluster.

:param snapshot_id: The ID to give the created snapshot.
:param cluster_id: The DB cluster to snapshot.
:return: Data about the newly created snapshot.
"""
    try:
        response = self.rds_client.create_db_cluster_snapshot(
            DBClusterSnapshotIdentifier=snapshot_id,
            DBClusterIdentifier=cluster_id)
        snapshot = response['DBClusterSnapshot']
    except ClientError as err:
        logger.error(
            "Couldn't create snapshot of %s. Here's why: %s: %s",
            cluster_id,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return snapshot
```

- For API details, see [CreateDBClusterSnapshot in AWS SDK for Python \(Boto3\) API Reference](#).

## Create a DB instance in a DB cluster

The following code example shows how to create a DB instance in an Aurora DB cluster.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
```

```
        return cls(rds_client)

    def create_instance_in_cluster(self, instance_id, cluster_id, db_engine,
instance_class):
        """
        Creates a database instance in an existing DB cluster. The first database that
is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is created.
        :param db_engine: The database engine of a database to create in the DB
instance.
                This must be compatible with the configured parameter group
of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id, DBClusterIdentifier=cluster_id,
                Engine=db_engine, DBInstanceClass=instance_class)
            db_inst = response['DBInstance']
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
        else:
            return db_inst
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a DB cluster

The following code example shows how to delete an Aurora DB cluster.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def delete_db_cluster(self, cluster_name):
```

```
"""
Deletes a DB cluster.

:param cluster_name: The name of the DB cluster to delete.
"""
try:
    self.rds_client.delete_db_cluster(
        DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True)
    logger.info("Deleted DB cluster %s.", cluster_name)
except ClientError:
    logger.exception("Couldn't delete DB cluster %s.", cluster_name)
    raise
```

- For API details, see [DeleteDBCluster in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete a DB cluster parameter group

The following code example shows how to delete an Aurora DB cluster parameter group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

- For API details, see [DeleteDBClusterParameterGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a DB instance

The following code example shows how to delete an Aurora DB instance.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:  
    """Encapsulates Aurora DB cluster actions."""  
    def __init__(self, rds_client):  
        """  
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)  
        client.  
        """  
        self.rds_client = rds_client  
  
    @classmethod  
    def from_client(cls):  
        """  
        Instantiates this class from a Boto3 client.  
        """  
        rds_client = boto3.client('rds')  
        return cls(rds_client)  
  
    def delete_db_instance(self, instance_id):  
        """  
        Deletes a DB instance.  
  
        :param instance_id: The ID of the DB instance to delete.  
        :return: Data about the deleted DB instance.  
        """  
        try:  
            response = self.rds_client.delete_db_instance(  
                DBInstanceIdentifier=instance_id, SkipFinalSnapshot=True,  
                DeleteAutomatedBackups=True)  
            db_inst = response['DBInstance']  
        except ClientError as err:  
            logger.error(  
                "Couldn't delete DB instance %s. Here's why: %s: %s", instance_id,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return db_inst
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe DB cluster parameter groups

The following code example shows how to describe Aurora DB cluster parameter groups.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name)
            parameter_group = response['DBClusterParameterGroups'][0]
        except ClientError as err:
            if err.response['Error']['Code'] == 'DBParameterGroupNotFound':
                logger.info("Parameter group %s does not exist.", parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response['Error']['Code'], err.response['Error']['Message'])
                raise
        else:
            return parameter_group
```

- For API details, see [DescribeDBClusterParameterGroups in AWS SDK for Python \(Boto3\) API Reference](#).

## Describe DB cluster snapshots

The following code example shows how to describe Aurora DB cluster snapshots.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
```

```
"""
    self.rds_client = rds_client

@classmethod
def from_client(cls):
"""
    Instantiates this class from a Boto3 client.
"""
    rds_client = boto3.client('rds')
    return cls(rds_client)

def get_cluster_snapshot(self, snapshot_id):
"""
    Gets a DB cluster snapshot.

:param snapshot_id: The ID of the snapshot to retrieve.
:return: The retrieved snapshot.
"""
try:
    response = self.rds_client.describe_db_cluster_snapshots(
        DBClusterSnapshotIdentifier=snapshot_id)
    snapshot = response['DBClusterSnapshots'][0]
except ClientError as err:
    logger.error(
        "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return snapshot
```

- For API details, see [DescribeDBClusterSnapshots](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe DB clusters

The following code example shows how to describe Aurora DB clusters.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_db_cluster(self, cluster_name):
        """
```

```
Gets data about an Aurora DB cluster.

:param cluster_name: The name of the DB cluster to retrieve.
:return: The retrieved DB cluster.
"""
try:
    response = self.rds_client.describe_db_clusters(
        DBClusterIdentifier=cluster_name)
    cluster = response['DBClusters'][0]
except ClientError as err:
    if err.response['Error']['Code'] == 'DBClusterNotFoundFault':
        logger.info("Cluster %s does not exist.", cluster_name)
    else:
        logger.error(
            "Couldn't verify the existence of DB cluster %s. Here's why: %s: %s",
            cluster_name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
else:
    return cluster
```

- For API details, see [DescribeDBClusters](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe DB instances

The following code example shows how to describe Aurora DB instances.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id)
            db_inst = response['DBInstances'][0]
```

```
        except ClientError as err:
            if err.response['Error']['Code'] == 'DBInstanceNotFound':
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response['Error']['Code'],
                    err.response['Error']['Message'])
                raise
        else:
            return db_inst
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe database engine versions

The following code example shows how to describe Aurora database engine versions.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned list of
                                       engine versions to those that are compatible
                                       with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {'Engine': engine}
            if parameter_group_family is not None:
                kwargs['DBParameterGroupFamily'] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response['DBEngineVersions']
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
```

```
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return versions
```

- For API details, see [DescribeDBEngineVersions in AWS SDK for Python \(Boto3\) API Reference](#).

## Describe options for DB instances

The following code example shows how to describe options for Aurora DB instances.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
        instance.
        :param db_engine_version: The engine version that must be supported by the DB
        instance.
        :return: The list of DB instance options that can be used to create a
        compatible DB instance.
        """
        try:
            inst_opts = []
            paginator =
                self.rds_client.get_paginator('describe_orderable_db_instance_options')
            for page in paginator.paginate(Engine=db_engine,
                                           EngineVersion=db_engine_version):
                inst_opts += page['OrderableDBInstanceOptions']
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return inst_opts
```

- For API details, see [DescribeOrderableDBInstanceOptions](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe parameters from a DB cluster parameter group

The following code example shows how to describe parameters from an Aurora DB cluster parameter group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:  
    """Encapsulates Aurora DB cluster actions."""  
    def __init__(self, rds_client):  
        """  
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)  
        client.  
        """  
        self.rds_client = rds_client  
  
    @classmethod  
    def from_client(cls):  
        """  
        Instantiates this class from a Boto3 client.  
        """  
        rds_client = boto3.client('rds')  
        return cls(rds_client)  
  
    def get_parameters(self, parameter_group_name, name_prefix='', source=None):  
        """  
        Gets the parameters that are contained in a DB cluster parameter group.  
  
        :param parameter_group_name: The name of the parameter group to query.  
        :param name_prefix: When specified, the retrieved list of parameters is  
        filtered  
            to contain only parameters that start with this prefix.  
        :param source: When specified, only parameters from this source are retrieved.  
            For example, a source of 'user' retrieves only parameters that  
            were set by a user.  
        :return: The list of requested parameters.  
        """  
        try:  
            kwargs = {'DBClusterParameterGroupName': parameter_group_name}  
            if source is not None:  
                kwargs['Source'] = source  
            parameters = []  
            paginator = self.rds_client.getPaginator('describe_db_cluster_parameters')  
            for page in paginator.paginate(**kwargs):  
                parameters += [  
                    p for p in page['Parameters'] if  
                    p['ParameterName'].startswith(name_prefix)]  
            except ClientError as err:  
                logger.error(  
                    "Couldn't get parameters for %s. Here's why: %s: %s",  
                    parameter_group_name,  
                    err.response['Error']['Code'], err.response['Error']['Message'])  
                raise  
        else:  
            return parameters
```

- For API details, see [DescribeDBClusterParameters](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update parameters in a DB cluster parameter group

The following code example shows how to update parameters in an Aurora DB cluster parameter group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AuroraWrapper:  
    """Encapsulates Aurora DB cluster actions."""  
    def __init__(self, rds_client):  
        """  
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)  
        client.  
        """  
        self.rds_client = rds_client  
  
    @classmethod  
    def from_client(cls):  
        """  
        Instantiates this class from a Boto3 client.  
        """  
        rds_client = boto3.client('rds')  
        return cls(rds_client)  
  
    def update_parameters(self, parameter_group_name, update_parameters):  
        """  
        Updates parameters in a custom DB cluster parameter group.  
  
        :param parameter_group_name: The name of the parameter group to update.  
        :param update_parameters: The parameters to update in the group.  
        :return: Data about the modified parameter group.  
        """  
        try:  
            response = self.rds_client.modify_db_cluster_parameter_group(  
                DBClusterParameterGroupName=parameter_group_name,  
                Parameters=update_parameters)  
        except ClientError as err:  
            logger.error(  
                "Couldn't update parameters in %s. Here's why: %s: %s",  
                parameter_group_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return response
```

- For API details, see [ModifyDBClusterParameterGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Get started with DB clusters

The following code example shows how to:

- Create a custom Aurora DB cluster parameter group and set parameter values.
- Create a DB cluster that is configured to use the parameter group.
- Create a DB instance in the DB cluster that contains a database.
- Take a snapshot of the DB cluster.
- Delete the instance, DB cluster, and parameter group.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
class AuroraClusterScenario:  
    """Runs a scenario that shows how to get started using Aurora DB clusters."""  
    def __init__(self, aurora_wrapper):  
        """  
        :param aurora_wrapper: An object that wraps Aurora DB cluster actions.  
        """  
        self.aurora_wrapper = aurora_wrapper  
  
    def create_parameter_group(self, db_engine, parameter_group_name):  
        """  
        Shows how to get available engine versions for a specified database engine and  
        create a DB cluster parameter group that is compatible with a selected engine  
        family.  
  
        :param db_engine: The database engine to use as a basis.  
        :param parameter_group_name: The name given to the newly created parameter  
        group.  
        :return: The newly created parameter group.  
        """  
        print(f"Checking for an existing DB cluster parameter group named  
{parameter_group_name}.")  
        parameter_group = self.aurora_wrapper.get_parameter_group(parameter_group_name)  
        if parameter_group is None:  
            print(f"Getting available database engine versions for {db_engine}.")  
            engine_versions = self.aurora_wrapper.get_engine_versions(db_engine)  
            families = list({ver['DBParameterGroupFamily'] for ver in engine_versions})  
            family_index = q.choose("Which family do you want to use? ", families)  
            print(f"Creating a DB cluster parameter group.")  
            self.aurora_wrapper.create_parameter_group(  
                parameter_group_name, families[family_index], 'Example parameter  
group.')  
            parameter_group =  
            self.aurora_wrapper.get_parameter_group(parameter_group_name)  
            print(f"Parameter group {parameter_group['DBClusterParameterGroupName']}:")  
            pp(parameter_group)  
            print('-'*88)  
            return parameter_group  
  
    def set_user_parameters(self, parameter_group_name):  
        """  
        Shows how to get the parameters contained in a custom parameter group and  
        update some of the parameter values in the group.  
  
        :param parameter_group_name: The name of the parameter group to query and  
        modify.  
        """  
        print("Let's set some parameter values in your parameter group.")  
        auto_inc_parameters = self.aurora_wrapper.get_parameters(  
            parameter_group_name)
```

```

        parameter_group_name, name_prefix='auto_increment')
update_params = []
for auto_inc in auto_inc_parameters:
    if auto_inc['IsModifiable'] and auto_inc['DataType'] == 'integer':
        print(f"The {auto_inc['ParameterName']} parameter is described as:")
        print(f"\t{auto_inc['Description']}")
        param_range = auto_inc['AllowedValues'].split('-')
        auto_inc['ParameterValue'] = str(q.ask(
            f"Enter a value between {param_range[0]} and {param_range[1]}: ",
            q.is_int, q.in_range(int(param_range[0]), int(param_range[1]))))
        update_params.append(auto_inc)
self.aurora_wrapper.update_parameters(parameter_group_name, update_params)
print("You can get a list of parameters you've set by specifying a source of
'user'.")
user_parameters = self.aurora_wrapper.get_parameters(parameter_group_name,
source='user')
pp(user_parameters)
print('*'*88)

def create_cluster(self, cluster_name, db_engine, db_name, parameter_group):
    """
    Shows how to create an Aurora DB cluster that contains a database of a
specified
    type. The database is also configured to use a custom DB cluster parameter
group.

    :param cluster_name: The name given to the newly created DB cluster.
    :param db_engine: The engine of the created database.
    :param db_name: The name given to the created database.
    :param parameter_group: The parameter group that is associated with the DB
cluster.
    :return: The newly created DB cluster.
    """
    print("Checking for an existing DB cluster.")
    cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
    if cluster is None:
        admin_username = q.ask(
            "Enter an administrator user name for the database: ", q.non_empty)
        admin_password = q.ask(
            "Enter a password for the administrator (at least 8 characters): ",
            q.non_empty)
        engine_versions = self.aurora_wrapper.get_engine_versions(
            db_engine, parameter_group['DBParameterGroupFamily'])
        engine_choices = [ver['EngineVersion'] for ver in engine_versions]
        print("The available engines for your parameter group are:")
        engine_index = q.choose("Which engine do you want to use? ",
        engine_choices)
        print(f"Creating DB cluster {cluster_name} and database {db_name}.\n"
              f"The DB cluster is configured to use\n"
              f"your custom parameter group
{parameter_group['DBClusterParameterGroupName']}.\n"
              f"and selected engine {engine_choices[engine_index]}.\n"
              f"This typically takes several minutes.")
        cluster = self.aurora_wrapper.create_db_cluster(
            cluster_name, parameter_group['DBClusterParameterGroupName'], db_name,
            db_engine, engine_choices[engine_index], admin_username,
            admin_password)
        while cluster.get('Status') != 'available':
            wait(10)
            cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
            print("Cluster created and available.\n")
    print("Cluster data:")
    pp(cluster)
    print('*'*88)
    return cluster

```

```

def create_instance(self, cluster):
    """
    Shows how to create a DB instance in an existing Aurora DB cluster. A new DB
    cluster contains no DB instances, so you must add one. The first DB instance that is
    added to a DB cluster defaults to a read-write DB instance.

    :param cluster: The DB cluster where the DB instance is added.
    :return: The newly created DB instance.
    """
    print("Checking for an existing database instance.")
    cluster_name = cluster['DBClusterIdentifier']
    db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
    if db_inst is None:
        print("Let's create a database instance in your DB cluster.")
        print("First, choose a DB instance type:")
        inst_opts = self.aurora_wrapper.get_orderable_instances(
            cluster['Engine'], cluster['EngineVersion'])
        inst_choices = list({opt['DBInstanceClass'] for opt in inst_opts})
        inst_index = q.choose("Which DB instance class do you want to use? ",
        inst_choices)
        print(f"Creating a database instance. This typically takes several
minutes.")
        db_inst = self.aurora_wrapper.create_instance_in_cluster(
            cluster_name, cluster_name, cluster['Engine'],
            inst_choices[inst_index])
        while db_inst.get('DBInstanceState') != 'available':
            wait(10)
        db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
    print("Instance data:")
    pp(db_inst)
    print('-'*88)
    return db_inst

@staticmethod
def display_connection(cluster):
    """
    Displays connection information about an Aurora DB cluster and tips on how to
    connect to it.

    :param cluster: The DB cluster to display.
    """
    print("You can now connect to your database using your favorite MySql client.
\\n"
          "One way to connect is by using the 'mysql' shell on an Amazon EC2
instance\\n"
          "that is running in the same VPC as your database cluster. Pass the
endpoint,\\n"
          "port, and administrator user name to 'mysql' and enter your password\\n"
          "when prompted:\\n")
    print(f"\n\\tmysql -h {cluster['Endpoint']} -P {cluster['Port']} -u
{cluster['MasterUsername']} -p\\n")
    print("For more information, see the User Guide for Aurora:\\n"
          "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora.Connect")
    print('-'*88)

def create_snapshot(self, cluster_name):
    """
    Shows how to create a DB cluster snapshot and wait until it's available.

    :param cluster_name: The name of a DB cluster to snapshot.
    """
    if q.ask("Do you want to create a snapshot of your DB cluster (y/n)? ",
    q.is_yesno):

```

```
snapshot_id = f"[cluster_name]-{uuid.uuid4()}"
print(f"Creating a snapshot named {snapshot_id}. This typically takes a few
minutes.")
snapshot = self.aurora_wrapper.create_cluster_snapshot(snapshot_id,
cluster_name)
while snapshot.get('Status') != 'available':
    wait(10)
    snapshot = self.aurora_wrapper.get_cluster_snapshot(snapshot_id)
pp(snapshot)
print('-'*88)

def cleanup(self, db_inst, cluster, parameter_group):
"""
Shows how to clean up a DB instance, DB cluster, and DB cluster parameter
group.
Before the DB cluster parameter group can be deleted, all associated DB
instances and
DB clusters must first be deleted.

:param db_inst: The DB instance to delete.
:param cluster: The DB cluster to delete.
:param parameter_group: The DB cluster parameter group to delete.
"""
cluster_name = cluster['DBClusterIdentifier']
parameter_group_name = parameter_group['DBClusterParameterGroupName']
if q.ask(
        "\nDo you want to delete the database instance, DB cluster, and
parameter "
        "group (y/n)? ", q.is_yesno):
    print(f"Deleting database instance {db_inst['DBInstanceIdentifier']}")
    self.aurora_wrapper.delete_db_instance(db_inst['DBInstanceIdentifier'])
    print(f"Deleting database cluster {cluster_name}.")
    self.aurora_wrapper.delete_db_cluster(cluster_name)
    print("Waiting for the DB instance and DB cluster to delete.\n"
          "This typically takes several minutes.")
    while db_inst is not None or cluster is not None:
        wait(10)
        if db_inst is not None:
            db_inst =
self.aurora_wrapper.get_db_instance(db_inst['DBInstanceIdentifier'])
            if cluster is not None:
                cluster =
self.aurora_wrapper.get_db_cluster(cluster['DBClusterIdentifier'])
                print(f"Deleting parameter group {parameter_group_name}.")
                self.aurora_wrapper.delete_parameter_group(parameter_group_name)

def run_scenario(self, db_engine, parameter_group_name, cluster_name, db_name):
    print('-'*88)
    print("Welcome to the Amazon Relational Database Service (Amazon RDS) get
started\n"
          "with Aurora DB clusters demo.")
    print('-'*88)

    parameter_group = self.create_parameter_group(db_engine, parameter_group_name)
    self.set_user_parameters(parameter_group_name)
    cluster = self.create_cluster(cluster_name, db_engine, db_name,
parameter_group)
    db_inst = self.create_instance(cluster)
    self.display_connection(cluster)
    self.create_snapshot(cluster_name)
    self.cleanup(db_inst, cluster, parameter_group)

    print("\nThanks for watching!")
    print('-'*88)
```

```

if __name__ == '__main__':
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    try:
        scenario = AuroraClusterScenario(AuroraWrapper.from_client())
        scenario.run_scenario(
            'aurora-mysql', 'doc-example-cluster-parameter-group', 'doc-example-
aurora',
            'docexampleddb')
    except Exception:
        logging.exception("Something went wrong with the demo.")

```

Define functions that are called by the scenario to manage Aurora actions.

```

class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon RDS)
        client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name)
            parameter_group = response['DBClusterParameterGroups'][0]
        except ClientError as err:
            if err.response['Error']['Code'] == 'DBParameterGroupNotFound':
                logger.info("Parameter group %s does not exist.", parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response['Error']['Code'], err.response['Error']['Message'])
                raise
        else:
            return parameter_group

    def create_parameter_group(self, parameter_group_name, parameter_group_family,
                             description):
        """
        Creates a DB cluster parameter group that is based on the specified parameter
        group family.

        :param parameter_group_name: The name of the newly created parameter group.
        :param parameter_group_family: The family that is used as the basis of the new
                                      parameter group.
        :param description: A description given to the parameter group.
        """

```

```

    :return: Data about the newly created parameter group.
    """
try:
    response = self.rds_client.create_db_cluster_parameter_group(
        DBClusterParameterGroupName=parameter_group_name,
        DBParameterGroupFamily=parameter_group_family,
        Description=description)
except ClientError as err:
    logger.error(
        "Couldn't create parameter group %s. Here's why: %s: %s",
        parameter_group_name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response

def delete_parameter_group(self, parameter_group_name):
    """
    Deletes a DB cluster parameter group.

    :param parameter_group_name: The name of the parameter group to delete.
    :return: Data about the parameter group.
    """
try:
    response = self.rds_client.delete_db_cluster_parameter_group(
        DBClusterParameterGroupName=parameter_group_name)
except ClientError as err:
    logger.error(
        "Couldn't delete parameter group %s. Here's why: %s: %s",
        parameter_group_name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response

def get_parameters(self, parameter_group_name, name_prefix='', source=None):
    """
    Gets the parameters that are contained in a DB cluster parameter group.

    :param parameter_group_name: The name of the parameter group to query.
    :param name_prefix: When specified, the retrieved list of parameters is
    filtered
        to contain only parameters that start with this prefix.
    :param source: When specified, only parameters from this source are retrieved.
        For example, a source of 'user' retrieves only parameters that
        were set by a user.
    :return: The list of requested parameters.
    """
try:
    kwargs = {'DBClusterParameterGroupName': parameter_group_name}
    if source is not None:
        kwargs['Source'] = source
    parameters = []
    paginator = self.rds_client.getPaginator('describe_db_cluster_parameters')
    for page in paginator.paginate(**kwargs):
        parameters += [
            p for p in page['Parameters'] if
            p['ParameterName'].startswith(name_prefix)]
    except ClientError as err:
        logger.error(
            "Couldn't get parameters for %s. Here's why: %s: %s",
            parameter_group_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return parameters

```

```

def update_parameters(self, parameter_group_name, update_parameters):
    """
    Updates parameters in a custom DB cluster parameter group.

    :param parameter_group_name: The name of the parameter group to update.
    :param update_parameters: The parameters to update in the group.
    :return: Data about the modified parameter group.
    """
    try:
        response = self.rds_client.modify_db_cluster_parameter_group(
            DBClusterParameterGroupName=parameter_group_name,
            Parameters=update_parameters)
    except ClientError as err:
        logger.error(
            "Couldn't update parameters in %s. Here's why: %s: %s",
            parameter_group_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response

def get_db_cluster(self, cluster_name):
    """
    Gets data about an Aurora DB cluster.

    :param cluster_name: The name of the DB cluster to retrieve.
    :return: The retrieved DB cluster.
    """
    try:
        response = self.rds_client.describe_db_clusters(
            DBClusterIdentifier=cluster_name)
        cluster = response['DBClusters'][0]
    except ClientError as err:
        if err.response['Error']['Code'] == 'DBClusterNotFoundFault':
            logger.info("Cluster %s does not exist.", cluster_name)
        else:
            logger.error(
                "Couldn't verify the existence of DB cluster %s. Here's why: %s: %s",
                cluster_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
    else:
        return cluster

def create_db_cluster(
        self, cluster_name, parameter_group_name, db_name, db_engine,
        db_engine_version, admin_name, admin_password):
    """
    Creates a DB cluster that is configured to use the specified parameter group.
    The newly created DB cluster contains a database that uses the specified engine
    and
    engine version.

    :param cluster_name: The name of the DB cluster to create.
    :param parameter_group_name: The name of the parameter group to associate with
        the DB cluster.
    :param db_name: The name of the database to create.
    :param db_engine: The database engine of the database that is created, such as
        MySql.
    :param db_engine_version: The version of the database engine.
    :param admin_name: The user name of the database administrator.
    :param admin_password: The password of the database administrator.
    :return: The newly created DB cluster.
    """
    try:

```

```
        response = self.rds_client.create_db_cluster(
            DatabaseName=db_name,
            DBClusterIdentifier=cluster_name,
            DBClusterParameterGroupName=parameter_group_name,
            Engine=db_engine,
            EngineVersion=db_engine_version,
            MasterUsername=admin_name,
            MasterUserPassword=admin_password)
        cluster = response['DBCluster']
    except ClientError as err:
        logger.error(
            "Couldn't create database %s. Here's why: %s: %s",
            db_name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return cluster

def delete_db_cluster(self, cluster_name):
    """
    Deletes a DB cluster.

    :param cluster_name: The name of the DB cluster to delete.
    """
    try:
        self.rds_client.delete_db_cluster(
            DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True)
        logger.info("Deleted DB cluster %s.", cluster_name)
    except ClientError:
        logger.exception("Couldn't delete DB cluster %s.", cluster_name)
        raise

def create_cluster_snapshot(self, snapshot_id, cluster_id):
    """
    Creates a snapshot of a DB cluster.

    :param snapshot_id: The ID to give the created snapshot.
    :param cluster_id: The DB cluster to snapshot.
    :return: Data about the newly created snapshot.
    """
    try:
        response = self.rds_client.create_db_cluster_snapshot(
            DBClusterSnapshotIdentifier=snapshot_id,
            DBClusterIdentifier=cluster_id)
        snapshot = response['DBClusterSnapshot']
    except ClientError as err:
        logger.error(
            "Couldn't create snapshot of %s. Here's why: %s: %s",
            cluster_id,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return snapshot

def get_cluster_snapshot(self, snapshot_id):
    """
    Gets a DB cluster snapshot.

    :param snapshot_id: The ID of the snapshot to retrieve.
    :return: The retrieved snapshot.
    """
    try:
        response = self.rds_client.describe_db_cluster_snapshots(
            DBClusterSnapshotIdentifier=snapshot_id)
        snapshot = response['DBClusterSnapshots'][0]
    except ClientError as err:
        logger.error(
            "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
            snapshot_id,
```

```

        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return snapshot

def create_instance_in_cluster(self, instance_id, cluster_id, db_engine,
instance_class):
    """
    Creates a database instance in an existing DB cluster. The first database that
is
    created defaults to a read-write DB instance.

:param instance_id: The ID to give the newly created DB instance.
:param cluster_id: The ID of the DB cluster where the DB instance is created.
:param db_engine: The database engine of a database to create in the DB
instance.
    This must be compatible with the configured parameter group
    of the DB cluster.
:param instance_class: The DB instance class for the newly created DB instance.
:return: Data about the newly created DB instance.
"""
try:
    response = self.rds_client.create_db_instance(
        DBInstanceIdentifier=instance_id, DBClusterIdentifier=cluster_id,
        Engine=db_engine, DBInstanceClass=instance_class)
    db_inst = response['DBInstance']
except ClientError as err:
    logger.error(
        "Couldn't create DB instance %s. Here's why: %s: %s",
        instance_id,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return db_inst

def get_engine_versions(self, engine, parameter_group_family=None):
    """
    Gets database engine versions that are available for the specified engine
and parameter group family.

:param engine: The database engine to look up.
:param parameter_group_family: When specified, restricts the returned list of
    engine versions to those that are compatible
with
    this parameter group family.
:return: The list of database engine versions.
"""
try:
    kwargs = {'Engine': engine}
    if parameter_group_family is not None:
        kwargs['DBParameterGroupFamily'] = parameter_group_family
    response = self.rds_client.describe_db_engine_versions(**kwargs)
    versions = response['DBEngineVersions']
except ClientError as err:
    logger.error(
        "Couldn't get engine versions for %s. Here's why: %s: %s",
        engine,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return versions

def get_orderable_instances(self, db_engine, db_engine_version):
    """
    Gets DB instance options that can be used to create DB instances that are
compatible with a set of specifications.

```

```
:param db_engine: The database engine that must be supported by the DB
instance.
:param db_engine_version: The engine version that must be supported by the DB
instance.
:return: The list of DB instance options that can be used to create a
compatible DB instance.
"""
try:
    inst_opts = []
    paginator =
self.rds_client.get_paginator('describe_orderable_db_instance_options')
    for page in paginator.paginate(Engine=db_engine,
EngineVersion=db_engine_version):
        inst_opts += page['OrderableDBInstanceOptions']
except ClientError as err:
    logger.error(
        "Couldn't get orderable DB instances. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return inst_opts

def get_db_instance(self, instance_id):
"""
Gets data about a DB instance.

:param instance_id: The ID of the DB instance to retrieve.
:return: The retrieved DB instance.
"""
try:
    response = self.rds_client.describe_db_instances(
        DBInstanceIdentifier=instance_id)
    db_inst = response['DBInstances'][0]
except ClientError as err:
    if err.response['Error']['Code'] == 'DBInstanceNotFound':
        logger.info("Instance %s does not exist.", instance_id)
    else:
        logger.error(
            "Couldn't get DB instance %s. Here's why: %s: %s",
            instance_id, err.response['Error']['Code'], err.response['Error']['Message'])
        raise
else:
    return db_inst

def delete_db_instance(self, instance_id):
"""
Deletes a DB instance.

:param instance_id: The ID of the DB instance to delete.
:return: Data about the deleted DB instance.
"""
try:
    response = self.rds_client.delete_db_instance(
        DBInstanceIdentifier=instance_id, SkipFinalSnapshot=True,
        DeleteAutomatedBackups=True)
    db_inst = response['DBInstance']
except ClientError as err:
    logger.error(
        "Couldn't delete DB instance %s. Here's why: %s: %s",
        instance_id, err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return db_inst
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateDBCluster](#)
  - [CreateDBClusterParameterGroup](#)
  - [CreateDBClusterSnapshot](#)
  - [CreateDBInstance](#)
  - [DeleteDBCluster](#)
  - [DeleteDBClusterParameterGroup](#)
  - [DeleteDBInstance](#)
  - [DescribeDBClusterParameterGroups](#)
  - [DescribeDBClusterParameters](#)
  - [DescribeDBClusterSnapshots](#)
  - [DescribeDBClusters](#)
  - [DescribeDBEngineVersions](#)
  - [DescribeDBInstances](#)
  - [DescribeOrderableDBInstanceStateOptions](#)
  - [ModifyDBClusterParameterGroup](#)

## CloudFront examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon CloudFront.

**Actions** are code excerpts that show you how to call individual service functions.

**Scenarios** are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3799\)](#)

## Actions

### Get distribution configuration

The following code example shows how to get Amazon CloudFront distribution configuration.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudFrontWrapper:  
    """Encapsulates Amazon CloudFront operations."""  
    def __init__(self, cloudfront_client):  
        """  
        :param cloudfront_client: A Boto3 CloudFront client  
        """  
        self.cloudfront_client = cloudfront_client  
  
    def update_distribution(self):  
        distribution_id = input(  
            "Enter the ID of the distribution you want to update: ")  
        # ...  
        # Call the CloudFront client to update the distribution  
        # ...  
        print(f"Updated distribution {distribution_id} successfully!")
```

```
"This script updates the comment for a CloudFront distribution.\n"
"Enter a CloudFront distribution ID: ")

distribution_config_response = self.cloudfront_client.get_distribution_config(
    Id=distribution_id)
distribution_config = distribution_config_response['DistributionConfig']
distribution_etag = distribution_config_response['ETag']

distribution_config['Comment'] = input(
    f"\nThe current comment for distribution {distribution_id} is "
    f"'{distribution_config['Comment']}'.\n"
    f"Enter a new comment: ")
self.cloudfront_client.update_distribution(
    DistributionConfig=distribution_config, Id=distribution_id,
    IfMatch=distribution_etag)
print("Done!")
```

- For API details, see [GetDistributionConfig](#) in *AWS SDK for Python (Boto3) API Reference*.

## List distributions

The following code example shows how to list Amazon CloudFront distributions.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""
    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client

    def list_distributions(self):
        print("CloudFront distributions:\n")
        distributions = self.cloudfront_client.list_distributions()
        if distributions['DistributionList']['Quantity'] > 0:
            for distribution in distributions['DistributionList']['Items']:
                print(f"Domain: {distribution['DomainName']}")
                print(f"Distribution Id: {distribution['Id']}")
                print(f"Certificate Source: "
                      f"{distribution['ViewerCertificate']['CertificateSource']}")
                if distribution['ViewerCertificate']['CertificateSource'] == "acm":
                    print(f"Certificate: {distribution['ViewerCertificate']}"
                          ['Certificate'])
                else:
                    print("")
        else:
            print("No CloudFront distributions detected.")
```

- For API details, see [ListDistributions](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update a distribution

The following code example shows how to update an Amazon CloudFront distribution.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""
    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client

    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: ")

        distribution_config_response = self.cloudfront_client.get_distribution_config(
            Id=distribution_id)
        distribution_config = distribution_config_response['DistributionConfig']
        distribution_etag = distribution_config_response['ETag']

        distribution_config['Comment'] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: ")
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config, Id=distribution_id,
            IfMatch=distribution_etag)
        print("Done!")
```

- For API details, see [UpdateDistribution](#) in *AWS SDK for Python (Boto3) API Reference*.

## CloudWatch examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon CloudWatch.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3801\)](#)
- [Scenarios \(p. 3808\)](#)

## Actions

### Create an alarm that watches a metric

The following code example shows how to create an Amazon CloudWatch alarm that watches a metric.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""
    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource

    def create_metric_alarm(
            self, metric_namespace, metric_name, alarm_name, stat_type, period,
            eval_periods, threshold, comparison_op):
        """
        Creates an alarm that watches a metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
        :param period: The period in which metric data are grouped to calculate
                       statistics.
        :param eval_periods: The number of periods that the metric must be over the
                            alarm threshold before the alarm is set into an alarmed
                            state.
        :param threshold: The threshold value to compare against the metric statistic.
        :param comparison_op: The comparison operation used to compare the threshold
                              against the metric.
        :return: The newly created alarm.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
            alarm = metric.put_alarm(
                AlarmName=alarm_name,
                Statistic=stat_type,
                Period=period,
                EvaluationPeriods=eval_periods,
                Threshold=threshold,
                ComparisonOperator=comparison_op)
            logger.info(
                "Added alarm %s to track metric %s.%s.", alarm_name, metric_namespace,
                metric_name)
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s", alarm_name, metric_namespace,
                metric_name)
            raise
        else:
            return alarm
```

- For API details, see [PutMetricAlarm in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete alarms

The following code example shows how to delete Amazon CloudWatch alarms.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:  
    """Encapsulates Amazon CloudWatch functions."""  
    def __init__(self, cloudwatch_resource):  
        """  
        :param cloudwatch_resource: A Boto3 CloudWatch resource.  
        """  
        self.cloudwatch_resource = cloudwatch_resource  
  
    def delete_metric_alarms(self, metric_namespace, metric_name):  
        """  
        Deletes all of the alarms that are currently watching the specified metric.  
  
        :param metric_namespace: The namespace of the metric.  
        :param metric_name: The name of the metric.  
        """  
        try:  
            metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)  
            metric.alarms.delete()  
            logger.info(  
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name)  
        except ClientError:  
            logger.exception(  
                "Couldn't delete alarms for metric %s.%s.", metric_namespace,  
                metric_name)  
            raise
```

- For API details, see [DeleteAlarms](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe alarms for a metric

The following code example shows how to describe Amazon CloudWatch alarms for a metric.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:  
    """Encapsulates Amazon CloudWatch functions."""  
    def __init__(self, cloudwatch_resource):  
        """  
        :param cloudwatch_resource: A Boto3 CloudWatch resource.  
        """  
        self.cloudwatch_resource = cloudwatch_resource  
  
    def get_metric_alarms(self, metric_namespace, metric_name):  
        """  
        Gets the alarms that are currently watching the specified metric.  
  
        :param metric_namespace: The namespace of the metric.  
        :param metric_name: The name of the metric.  
        :returns: An iterator that yields the alarms.  
        """
```

```
"""
metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
alarm_iter = metric.alarms.all()
logger.info("Got alarms for metric %s.%s.", metric_namespace, metric_name)
return alarm_iter
```

- For API details, see [DescribeAlarmsForMetric](#) in *AWS SDK for Python (Boto3) API Reference*.

## Disable alarm actions

The following code example shows how to disable Amazon CloudWatch alarm actions.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""
    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource

    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise, they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.", "Enabled" if enable else "Disabled",
                alarm_name)
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.", "enable" if enable else "disable",
                alarm_name)
        raise
```

- For API details, see [DisableAlarmActions](#) in *AWS SDK for Python (Boto3) API Reference*.

## Enable alarm actions

The following code example shows how to enable Amazon CloudWatch alarm actions.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""
    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource

    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise, they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.", "Enabled" if enable else "Disabled",
                alarm_name)
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.", "enable" if enable else "disable",
                alarm_name)
        raise
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get metric statistics

The following code example shows how to get Amazon CloudWatch metric statistics.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""
    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource
```

```
def get_metric_statistics(self, namespace, name, start, end, period, stat_types):
    """
    Gets statistics for a metric within a specified time span. Metrics are grouped
    into the specified period.

    :param namespace: The namespace of the metric.
    :param name: The name of the metric.
    :param start: The UTC start time of the time span to retrieve.
    :param end: The UTC end time of the time span to retrieve.
    :param period: The period, in seconds, in which to group metrics. The period
        must match the granularity of the metric, which depends on
        the metric's age. For example, metrics that are older than
        three hours have a one-minute granularity, so the period must
        be at least 60 and must be a multiple of 60.
    :param stat_types: The type of statistics to retrieve, such as average value
        or maximum value.
    :return: The retrieved statistics for the metric.
    """
    try:
        metric = self.cloudwatch_resource.Metric(namespace, name)
        stats = metric.get_statistics(
            StartTime=start, EndTime=end, Period=period, Statistics=stat_types)
        logger.info(
            "Got %s statistics for %s.", len(stats['Datapoints']), stats['Label'])
    except ClientError:
        logger.exception("Couldn't get statistics for %s.%s.", namespace, name)
        raise
    else:
        return stats
```

- For API details, see [GetMetricStatistics](#) in *AWS SDK for Python (Boto3) API Reference*.

## List metrics

The following code example shows how to list Amazon CloudWatch metrics.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:
    """
    Encapsulates Amazon CloudWatch functions.
    """
    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource

    def list_metrics(self, namespace, name, recent=False):
        """
        Gets the metrics within a namespace that have the specified name.
        If the metric has no dimensions, a single metric is returned.
        Otherwise, metrics for all dimensions are returned.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param recent: When True, only metrics that have been active in the last
            three hours are returned.
        :return: An iterator that yields the retrieved metrics.
        """
```

```
"""
try:
    kwargs = {'Namespace': namespace, 'MetricName': name}
    if recent:
        kwargs['RecentlyActive'] = 'PT3H' # List past 3 hours only
    metric_iter = self.cloudwatch_resource.metrics.filter(**kwargs)
    logger.info("Got metrics for %s.%s.", namespace, name)
except ClientError:
    logger.exception("Couldn't get metrics for %s.%s.", namespace, name)
    raise
else:
    return metric_iter
```

- For API details, see [ListMetrics](#) in *AWS SDK for Python (Boto3) API Reference*.

## Put a set of data into a metric

The following code example shows how to put a set of data into a Amazon CloudWatch metric.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""
    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource

    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
        :param data_set: The set of data to send. This set is a dictionary that
                        contains a list of values and a list of corresponding counts.
                        The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{
                    'MetricName': name,
                    'Timestamp': timestamp,
                    'Values': data_set['values'],
                    'Counts': data_set['counts'],
                    'Unit': unit}])
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.", namespace,
                             name)
            raise
```

- For API details, see [PutMetricData](#) in *AWS SDK for Python (Boto3) API Reference*.

## Put data into a metric

The following code example shows how to put data into a Amazon CloudWatch metric.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CloudWatchWrapper:  
    """Encapsulates Amazon CloudWatch functions."""  
    def __init__(self, cloudwatch_resource):  
        """  
        :param cloudwatch_resource: A Boto3 CloudWatch resource.  
        """  
        self.cloudwatch_resource = cloudwatch_resource  
  
    def put_metric_data(self, namespace, name, value, unit):  
        """  
        Sends a single data value to CloudWatch for a metric. This metric is given  
        a timestamp of the current UTC time.  
  
        :param namespace: The namespace of the metric.  
        :param name: The name of the metric.  
        :param value: The value of the metric.  
        :param unit: The unit of the metric.  
        """  
        try:  
            metric = self.cloudwatch_resource.Metric(namespace, name)  
            metric.put_data(  
                Namespace=namespace,  
                MetricData=[{  
                    'MetricName': name,  
                    'Value': value,  
                    'Unit': unit  
                }]  
            )  
            logger.info("Put data for metric %s.%s", namespace, name)  
        except ClientError:  
            logger.exception("Couldn't put data for metric %s.%s", namespace, name)  
            raise
```

- For API details, see [PutMetricData](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Manage Amazon CloudWatch metrics and alarms

The following code example shows how to:

- Create an alarm that watches a metric.
- Put data into a CloudWatch metric and trigger the alarm.
- Get data from the alarm.

- Delete the alarm.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps CloudWatch operations.

```
from datetime import datetime, timedelta
import logging
from pprint import pprint
import random
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""
    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource

    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
        :param data_set: The set of data to send. This set is a dictionary that
                        contains a list of values and a list of corresponding counts.
                        The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{
                    'MetricName': name,
                    'Timestamp': timestamp,
                    'Values': data_set['values'],
                    'Counts': data_set['counts'],
                    'Unit': unit}])
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.", namespace,
                             name)
            raise

    def create_metric_alarm(
            self, metric_namespace, metric_name, alarm_name, stat_type, period,
            eval_periods, threshold, comparison_op):
        """
        Creates an alarm that watches a metric.
        
```

```

:param metric_namespace: The namespace of the metric.
:param metric_name: The name of the metric.
:param alarm_name: The name of the alarm.
:param stat_type: The type of statistic the alarm watches.
:param period: The period in which metric data are grouped to calculate
    statistics.
:param eval_periods: The number of periods that the metric must be over the
    alarm threshold before the alarm is set into an alarmed
    state.
:param threshold: The threshold value to compare against the metric statistic.
:param comparison_op: The comparison operation used to compare the threshold
    against the metric.
:return: The newly created alarm.
"""
try:
    metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
    alarm = metric.put_alarm(
        AlarmName=alarm_name,
        Statistic=stat_type,
        Period=period,
        EvaluationPeriods=eval_periods,
        Threshold=threshold,
        ComparisonOperator=comparison_op)
    logger.info(
        "Added alarm %s to track metric %s.%s.", alarm_name, metric_namespace,
        metric_name)
except ClientError:
    logger.exception(
        "Couldn't add alarm %s to metric %s.%s", alarm_name, metric_namespace,
        metric_name)
    raise
else:
    return alarm

def put_metric_data(self, namespace, name, value, unit):
    """
    Sends a single data value to CloudWatch for a metric. This metric is given
    a timestamp of the current UTC time.

    :param namespace: The namespace of the metric.
    :param name: The name of the metric.
    :param value: The value of the metric.
    :param unit: The unit of the metric.
    """
    try:
        metric = self.cloudwatch_resource.Metric(namespace, name)
        metric.put_data(
            Namespace=namespace,
            MetricData=[{
                'MetricName': name,
                'Value': value,
                'Unit': unit
            }])
    )
    logger.info("Put data for metric %s.%s", namespace, name)
except ClientError:
    logger.exception("Couldn't put data for metric %s.%s", namespace, name)
    raise

def get_metric_statistics(self, namespace, name, start, end, period, stat_types):
    """
    Gets statistics for a metric within a specified time span. Metrics are grouped
    into the specified period.

    :param namespace: The namespace of the metric.
    :param name: The name of the metric.
    """

```

```

:param start: The UTC start time of the time span to retrieve.
:param end: The UTC end time of the time span to retrieve.
:param period: The period, in seconds, in which to group metrics. The period
               must match the granularity of the metric, which depends on
               the metric's age. For example, metrics that are older than
               three hours have a one-minute granularity, so the period must
               be at least 60 and must be a multiple of 60.
:param stat_types: The type of statistics to retrieve, such as average value
                   or maximum value.
:return: The retrieved statistics for the metric.
"""
try:
    metric = self.cloudwatch_resource.Metric(namespace, name)
    stats = metric.get_statistics(
        StartTime=start, EndTime=end, Period=period, Statistics=stat_types)
    logger.info(
        "Got %s statistics for %s.", len(stats['Datapoints']), stats['Label'])
except ClientError:
    logger.exception("Couldn't get statistics for %s.%s.", namespace, name)
    raise
else:
    return stats

def get_metric_alarms(self, metric_namespace, metric_name):
    """
    Gets the alarms that are currently watching the specified metric.

    :param metric_namespace: The namespace of the metric.
    :param metric_name: The name of the metric.
    :returns: An iterator that yields the alarms.
    """
    metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
    alarm_iter = metric.alarms.all()
    logger.info("Got alarms for metric %s.%s.", metric_namespace, metric_name)
    return alarm_iter

def delete_metric_alarms(self, metric_namespace, metric_name):
    """
    Deletes all of the alarms that are currently watching the specified metric.

    :param metric_namespace: The namespace of the metric.
    :param metric_name: The name of the metric.
    """
    try:
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        metric.alarms.delete()
        logger.info(
            "Deleted alarms for metric %s.%s.", metric_namespace, metric_name)
    except ClientError:
        logger.exception(
            "Couldn't delete alarms for metric %s.%s.", metric_namespace,
            metric_name)
        raise

```

Use the wrapper class to put data in a metric, trigger an alarm that watches the metric, and get data from the alarm.

```

def usage_demo():
    print('*'*88)
    print("Welcome to the Amazon CloudWatch metrics and alarms demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

```

```

cw_wrapper = CloudWatchWrapper(boto3.resource('cloudwatch'))

minutes = 20
metric_namespace = 'doc-example-metric'
metric_name = 'page_views'
start = datetime.utcnow() - timedelta(minutes=minutes)
print(f"Putting data into metric {metric_namespace}.{metric_name} spanning the "
      f"last {minutes} minutes.")
for offset in range(0, minutes):
    stamp = start + timedelta(minutes=offset)
    cw_wrapper.put_metric_data_set(
        metric_namespace, metric_name, stamp, 'Count', {
            'values': [
                random.randint(bound, bound * 2)
                for bound in range(offset + 1, offset + 11)],
            'counts': [random.randint(1, offset + 1) for _ in range(10)]})
)

alarm_name = 'high_page_views'
period = 60
eval_periods = 2
print(f"Creating alarm {alarm_name} for metric {metric_name}.")
alarm = cw_wrapper.create_metric_alarm(
    metric_namespace, metric_name, alarm_name, 'Maximum', period, eval_periods,
    100, 'GreaterThanThreshold')
print(f"Alarm ARN is {alarm.alarm_arn}.")
print(f"Current alarm state is: {alarm.state_value}.")

print(f"Sending data to trigger the alarm. This requires data over the threshold "
      f"for {eval_periods} periods of {period} seconds each.")
while alarm.state_value == 'INSUFFICIENT_DATA':
    print("Sending data for the metric.")
    cw_wrapper.put_metric_data(
        metric_namespace, metric_name, random.randint(100, 200), 'Count')
    alarm.load()
    print(f"Current alarm state is: {alarm.state_value}.")
    if alarm.state_value == 'INSUFFICIENT_DATA':
        print(f"Waiting for {period} seconds...")
        time.sleep(period)
    else:
        print("Wait for a minute for eventual consistency of metric data.")
        time.sleep(period)
    if alarm.state_value == 'OK':
        alarm.load()
        print(f"Current alarm state is: {alarm.state_value}.")

print(f"Getting data for metric {metric_namespace}.{metric_name} during timespan "
      f"of {start} to {datetime.utcnow()} (times are UTC).")
stats = cw_wrapper.get_metric_statistics(
    metric_namespace, metric_name, start, datetime.utcnow(), 60,
    ['Average', 'Minimum', 'Maximum'])
print(f"Got {len(stats['Datapoints'])} data points for metric "
      f"{metric_namespace}.{metric_name}.")
pprint(sorted(stats['Datapoints'], key=lambda x: x['Timestamp']))

print(f"Getting alarms for metric {metric_name}.")
alarms = cw_wrapper.get_metric_alarms(metric_namespace, metric_name)
for alarm in alarms:
    print(f"Alarm {alarm.name} is currently in state {alarm.state_value}.")

print(f"Deleting alarms for metric {metric_name}.")
cw_wrapper.delete_metric_alarms(metric_namespace, metric_name)

print("Thanks for watching!")
print('*'*88)

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [DeleteAlarms](#)
  - [DescribeAlarmsForMetric](#)
  - [DisableAlarmActions](#)
  - [EnableAlarmActions](#)
  - [GetMetricStatistics](#)
  - [ListMetrics](#)
  - [PutMetricAlarm](#)
  - [PutMetricData](#)

## Amazon Cognito Identity Provider examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Cognito Identity Provider.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3813\)](#)
- [Scenarios \(p. 3826\)](#)

## Actions

### Confirm a user

The following code example shows how to confirm an Amazon Cognito user.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:  
    """Encapsulates Amazon Cognito actions"""  
    def __init__(self, cognito_idp_client, user_pool_id, client_id,  
                 client_secret=None):  
        """  
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.  
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.  
        :param client_id: The ID of a client application registered with the user pool.  
        :param client_secret: The client secret, if the client has a secret.  
        """  
        self.cognito_idp_client = cognito_idp_client  
        self.user_pool_id = user_pool_id  
        self.client_id = client_id  
        self.client_secret = client_secret  
  
    def confirm_user_sign_up(self, user_name, confirmation_code):
```

```
"""
Confirms a previously created user. A user must be confirmed before they
can sign in to Amazon Cognito.

:param user_name: The name of the user to confirm.
:param confirmation_code: The confirmation code sent to the user's registered
                           email address.
:return: True when the confirmation succeeds.
"""
try:
    kwargs = {
        'ClientId': self.client_id, 'Username': user_name,
        'ConfirmationCode': confirmation_code}
    if self.client_secret is not None:
        kwargs['SecretHash'] = self._secret_hash(user_name)
    self.cognito_idp_client.confirm_sign_up(**kwargs)
except ClientError as err:
    logger.error(
        "Couldn't confirm sign up for %s. Here's why: %s: %s",
        user_name, err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return True
```

- For API details, see [ConfirmSignUp in AWS SDK for Python \(Boto3\) API Reference](#).

## Confirm an MFA device for tracking

The following code example shows how to confirm an MFA device for tracking by Amazon Cognito.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret

    def confirm_mfa_device(
            self, user_name, device_key, device_group_key, device_password,
            access_token,
            aws_srp):
        """
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a new
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        """
        pass
```

```

:param device_group_key: The group key of the device, returned by Amazon
Cognito.
:param device_password: The password that is associated with the device.
:param access_token: The user's access token.
:param aws_srp: A class that helps with Secure Remote Password (SRP)
calculations. The scenario associated with this example uses
the warrant package.
:return: True when the user must confirm the device. Otherwise, False. When
False, the device is automatically confirmed and tracked.
"""
srp_helper = aws_srp.AWSSRP(
    username=user_name, password=device_password,
    pool_id='_', client_id=self.client_id, client_secret=None,
    client=self.cognito_idp_client)
device_and_pw = f'{device_group_key}{device_key}:{device_password}'
device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode('utf-8'))
salt = aws_srp.pad_hex(aws_srp.get_random(16))
x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt + device_and_pw_hash))
verifier = aws_srp.pad_hex(pow(srp_helper.g, x_value, srp_helper.big_n))
device_secret_verifier_config = {
    "PasswordVerifier":
base64.standard_b64encode(bytarray.fromhex(verifier)).decode('utf-8'),
    "Salt": base64.standard_b64encode(bytarray.fromhex(salt)).decode('utf-8')
}
try:
    response = self.cognito_idp_client.confirm_device(
        AccessToken=access_token,
        DeviceKey=device_key,
        DeviceSecretVerifierConfig=device_secret_verifier_config)
    user_confirm = response['UserConfirmationNecessary']
except ClientError as err:
    logger.error(
        "Couldn't confirm mfa device %s. Here's why: %s: %s",
        device_key,
        err.response['Error']['Code'],
        err.response['Error']['Message'])
    raise
else:
    return user_confirm

```

- For API details, see [ConfirmDevice](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a token to associate an MFA application with a user

The following code example shows how to get a token to associate an MFA application with an Amazon Cognito user.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user pool.
        :param client_secret: The client secret, if the client has a secret.
        """

```

```
    self.cognito_idp_client = cognito_idp_client
    self.user_pool_id = user_pool_id
    self.client_id = client_id
    self.client_secret = client_secret

    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the user.

        :param session: Session information returned from a previous call to initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
        self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            response.pop('ResponseMetadata', None)
        return response
```

- For API details, see [AssociateSoftwareToken](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get information about a user

The following code example shows how to get information about an Amazon Cognito user.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret

    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon Cognito
        to send an email to the specified email address. The email contains a code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        """

```

```
:param password: The password for the new user.
:param user_email: The email address for the new user.
:return: True when the user is already confirmed with Amazon Cognito.
        Otherwise, false.
"""
try:
    kwargs = {
        'ClientId': self.client_id, 'Username': user_name, 'Password': password,
        'UserAttributes': [{'Name': 'email', 'Value': user_email}]}
    if self.client_secret is not None:
        kwargs['SecretHash'] = self._secret_hash(user_name)
    response = self.cognito_idp_client.sign_up(**kwargs)
    confirmed = response['UserConfirmed']
except ClientError as err:
    if err.response['Error']['Code'] == 'UsernameExistsException':
        response = self.cognito_idp_client.admin_get_user(
            UserPoolId=self.user_pool_id, Username=user_name)
        logger.warning("User %s exists and is %s.", user_name,
response['UserStatus'])
        confirmed = response['UserStatus'] == 'CONFIRMED'
    else:
        logger.error(
            "Couldn't sign up %s. Here's why: %s: %s",
            user_name, err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
return confirmed
```

- For API details, see [Admin GetUser](#) in *AWS SDK for Python (Boto3) API Reference*.

## List users

The following code example shows how to list Amazon Cognito users.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret

    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
```

```
        response = self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
        users = response['Users']
    except ClientError as err:
        logger.error(
            "Couldn't list users for %s. Here's why: %s: %s",
            self.user_pool_id,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return users
```

- For API details, see [ListUsers](#) in *AWS SDK for Python (Boto3) API Reference*.

## Resend a confirmation code

The following code example shows how to resend an Amazon Cognito confirmation code.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret

    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {
                'ClientId': self.client_id, 'Username': user_name}
            if self.client_secret is not None:
                kwargs['SecretHash'] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response['CodeDeliveryDetails']
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
        else:
            return delivery
```

- For API details, see [ResendConfirmationCode](#) in *AWS SDK for Python (Boto3) API Reference*.

## Respond to SRP authentication challenges

The following code example shows how to respond to Amazon Cognito SRP authentication challenges.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Sign in with a tracked device. To complete sign-in, the client must respond correctly to Secure Remote Password (SRP) challenges.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret

    def sign_in_with_tracked_device(
            self, user_name, password, device_key, device_group_key, device_password,
            aws_srp):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in with a tracked device lets a user sign in without entering a new MFA code.

        Signing in with a tracked device requires that the client respond to the SRP protocol. The scenario associated with this example uses the warrant package to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/Secure\_Remote\_Password\_protocol.
        """
        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains an access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name, password=device_password,
                pool_id='__', client_id=self.client_id, client_secret=None,
                client=self.cognito_idp_client)

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id, AuthFlow='USER_PASSWORD_AUTH',
                AuthParameters={
                    'USERNAME': user_name, 'PASSWORD': password, 'DEVICE_KEY':
device_key})
            if response_init['ChallengeName'] != 'DEVICE_SRP_AUTH':
                raise RuntimeError(

```

```
f"Expected DEVICE_SRP_AUTH challenge but got {response_init['ChallengeName']}.")

auth_params = srp_helper.get_auth_params()
auth_params['DEVICE_KEY'] = device_key
response_auth = self.cognito_idp_client.respond_to_auth_challenge(
    ClientId=self.client_id,
    ChallengeName='DEVICE_SRP_AUTH',
    ChallengeResponses=auth_params
)
if response_auth['ChallengeName'] != 'DEVICE_PASSWORD_VERIFIER':
    raise RuntimeError(
        f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
        f"{response_init['ChallengeName']}.")

challenge_params = response_auth['ChallengeParameters']
challenge_params['USER_ID_FOR_SRP'] = device_group_key + device_key
cr = srp_helper.process_challenge(challenge_params)
cr['USERNAME'] = user_name
cr['DEVICE_KEY'] = device_key
response_verifier = self.cognito_idp_client.respond_to_auth_challenge(
    ClientId=self.client_id,
    ChallengeName='DEVICE_PASSWORD_VERIFIER',
    ChallengeResponses=cr)
auth_tokens = response_verifier['AuthenticationResult']
except ClientError as err:
    logger.error(
        "Couldn't start client sign in for %s. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return auth_tokens
```

- For API details, see [RespondToAuthChallenge in AWS SDK for Python \(Boto3\) API Reference](#).

## Respond to an authentication challenge

The following code example shows how to respond to an Amazon Cognito authentication challenge.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Respond to an MFA challenge by providing a code generated by an associated MFA application.

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret
```

```

def respond_to_mfa_challenge(self, user_name, session, mfa_code):
    """
    Responds to a challenge for an MFA code. This completes the second step of
    a two-factor sign-in. When sign-in is successful, it returns an access token
    that can be used to get AWS credentials from Amazon Cognito.

    :param user_name: The name of the user who is signing in.
    :param session: Session information returned from a previous call to initiate
                    authentication.
    :param mfa_code: A code generated by the associated MFA application.
    :return: The result of the authentication. When successful, this contains an
            access token for the user.
    """
    try:
        kwargs = {
            'UserPoolId': self.user_pool_id,
            'ClientId': self.client_id,
            'ChallengeName': 'SOFTWARE_TOKEN_MFA', 'Session': session,
            'ChallengeResponses': {
                'USERNAME': user_name, 'SOFTWARE_TOKEN_MFA_CODE': mfa_code}}
        if self.client_secret is not None:
            kwargs['ChallengeResponses']['SECRET_HASH'] =
self._secret_hash(user_name)
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response['AuthenticationResult']
        except ClientError as err:
            if err.response['Error']['Code'] == 'ExpiredCodeException':
                logger.warning(
                    "Your MFA code has expired or has been used already. You might have"
                    "to wait a few seconds until your app shows you a new code.")
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s: %s",
                    user_name,
                    err.response['Error']['Code'], err.response['Error']['Message'])
                raise
        else:
            return auth_result
    
```

- For API details, see [AdminRespondToAuthChallenge](#) in *AWS SDK for Python (Boto3) API Reference*.

## Sign up a user

The following code example shows how to sign up a user with Amazon Cognito.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class CognitoIdentityProviderWrapper:
    """
    Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user pool.
    
```

```
:param client_secret: The client secret, if the client has a secret.  
"""  
    self.cognito_idp_client = cognito_idp_client  
    self.user_pool_id = user_pool_id  
    self.client_id = client_id  
    self.client_secret = client_secret  
  
def sign_up_user(self, user_name, password, user_email):  
    """  
        Signs up a new user with Amazon Cognito. This action prompts Amazon Cognito  
        to send an email to the specified email address. The email contains a code that  
        can be used to confirm the user.  
  
        When the user already exists, the user status is checked to determine whether  
        the user has been confirmed.  
  
        :param user_name: The user name that identifies the new user.  
        :param password: The password for the new user.  
        :param user_email: The email address for the new user.  
        :return: True when the user is already confirmed with Amazon Cognito.  
                Otherwise, false.  
    """  
  
    try:  
        kwargs = {  
            'ClientId': self.client_id, 'Username': user_name, 'Password':  
password,  
            'UserAttributes': [{'Name': 'email', 'Value': user_email}]}  
        if self.client_secret is not None:  
            kwargs['SecretHash'] = self._secret_hash(user_name)  
        response = self.cognito_idp_client.sign_up(**kwargs)  
        confirmed = response['UserConfirmed']  
    except ClientError as err:  
        if err.response['Error']['Code'] == 'UsernameExistsException':  
            response = self.cognito_idp_client.admin_get_user(  
                UserPoolId=self.user_pool_id, Username=user_name)  
            logger.warning("User %s exists and is %s.", user_name,  
response['UserStatus'])  
            confirmed = response['UserStatus'] == 'CONFIRMED'  
        else:  
            logger.error(  
                "Couldn't sign up %s. Here's why: %s: %s", user_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
    return confirmed
```

- For API details, see [SignUp in AWS SDK for Python \(Boto3\) API Reference](#).

## Start authentication with a tracked device

The following code example shows how to start authentication with a device tracked by Amazon Cognito.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Sign in with a tracked device. To complete sign-in, the client must respond correctly to Secure Remote Password (SRP) challenges.

```
class CognitoIdentityProviderWrapper:
```

```
"""Encapsulates Amazon Cognito actions"""
def __init__(self, cognito_idp_client, user_pool_id, client_id,
client_secret=None):
    """
    :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
    :param user_pool_id: The ID of an existing Amazon Cognito user pool.
    :param client_id: The ID of a client application registered with the user pool.
    :param client_secret: The client secret, if the client has a secret.
    """
    self.cognito_idp_client = cognito_idp_client
    self.user_pool_id = user_pool_id
    self.client_id = client_id
    self.client_secret = client_secret

    def sign_in_with_tracked_device(
        self, user_name, password, device_key, device_group_key, device_password,
        aws_srp):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA code.

        Signing in with a tracked device requires that the client respond to the SRP
        protocol. The scenario associated with this example uses the warrant package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/Secure\_Remote\_Password\_protocol.
        """
        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
            associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains an
            access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name, password=device_password,
                pool_id='__', client_id=self.client_id, client_secret=None,
                client=self.cognito_idp_client)

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id, AuthFlow='USER_PASSWORD_AUTH',
                AuthParameters={
                    'USERNAME': user_name, 'PASSWORD': password, 'DEVICE_KEY':
                    device_key})
            if response_init['ChallengeName'] != 'DEVICE_SRP_AUTH':
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got "
                    f"{response_init['ChallengeName']}.")

            auth_params = srp_helper.get_auth_params()
            auth_params['DEVICE_KEY'] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName='DEVICE_SRP_AUTH',
                ChallengeResponses=auth_params
            )
            if response_auth['ChallengeName'] != 'DEVICE_PASSWORD_VERIFIER':
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_auth['ChallengeName']}.")

        except Exception as e:
            raise RuntimeError(f"Error during sign-in: {e}")
    
```

```
challenge_params = response_auth['ChallengeParameters']
challenge_params['USER_ID_FOR_SRP'] = device_group_key + device_key
cr = srp_helper.process_challenge(challenge_params)
cr['USERNAME'] = user_name
cr['DEVICE_KEY'] = device_key
response_verifier = self.cognito_idp_client.respond_to_auth_challenge(
    ClientId=self.client_id,
    ChallengeName='DEVICE_PASSWORD_VERIFIER',
    ChallengeResponses=cr)
auth_tokens = response_verifier['AuthenticationResult']
except ClientError as err:
    logger.error(
        "Couldn't start client sign in for %s. Here's why: %s: %s",
        user_name,
        err.response['Error']['Code'],
        err.response['Error']['Message'])
    raise
else:
    return auth_tokens
```

- For API details, see [InitiateAuth](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start authentication with administrator credentials

The following code example shows how to start authentication with Amazon Cognito and administrator credentials.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret

    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure server.

        If the user pool is configured to require MFA and this is the first sign-in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful, this
                returns an access token that can be used to get AWS credentials.
        Otherwise,
```

```

Amazon Cognito returns a challenge to set up an MFA application,
or a challenge to enter an MFA code from a registered MFA application.

"""

try:
    kwargs = {
        'UserPoolId': self.user_pool_id,
        'ClientId': self.client_id,
        'AuthFlow': 'ADMIN_USER_PASSWORD_AUTH',
        'AuthParameters': {'USERNAME': user_name, 'PASSWORD': password}}
    if self.client_secret is not None:
        kwargs['AuthParameters']['SECRET_HASH'] = self._secret_hash(user_name)
    response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
    challenge_name = response.get('ChallengeName', None)
    if challenge_name == 'MFA_SETUP':
        if 'SOFTWARE_TOKEN_MFA' in response['ChallengeParameters']
            ['MFAS_CAN_SETUP']:
            response.update(self.get_mfa_secret(response['Session']))
        else:
            raise RuntimeError(
                "The user pool requires MFA setup, but the user pool is not "
                "configured for TOTP MFA. This example requires TOTP MFA.")
    except ClientError as err:
        logger.error(
            "Couldn't start sign in for %s. Here's why: %s: %s",
            user_name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        response.pop('ResponseMetadata', None)
    return response

```

- For API details, see [AdminInitiateAuth](#) in *AWS SDK for Python (Boto3) API Reference*.

## Verify an MFA application with a user

The following code example shows how to verify an MFA application with an Amazon Cognito user.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""
    def __init__(self, cognito_idp_client, user_pool_id, client_id,
                 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret

    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

```

```
:param session: Session information returned from a previous call to initiate
    authentication.
:param user_code: A code generated by the associated MFA application.
:return: Status that indicates whether the MFA application is verified.
"""
try:
    response = self.cognito_idp_client.verify_software_token(
        Session=session, UserCode=user_code)
except ClientError as err:
    logger.error(
        "Couldn't verify MFA. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    response.pop('ResponseMetadata', None)
return response
```

- For API details, see [VerifySoftwareToken](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Sign up a user with a user pool that requires MFA

The following code example shows how to:

- Sign up a user with a user name, password, and email address.
- Confirm the user from a code sent in email.
- Set up multi-factor authentication by associating an MFA application with the user.
- Sign in by using a password and an MFA code.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

In addition to the previously listed steps, this example also registers an MFA device to be tracked by Amazon Cognito and shows you how to sign in by using a password and information from the tracked device. This avoids the need to enter a new MFA code.

```
def run_scenario(cognito_idp_client, user_pool_id, client_id):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the Amazon Cognito user signup with MFA demo.")
    print('*'*88)

    cog_wrapper = CognitoIdentityProviderWrapper(cognito_idp_client, user_pool_id,
                                                client_id)

    user_name = q.ask("Let's sign up a new user. Enter a user name: ", q.non_empty)
    password = q.ask("Enter a password for the user: ", q.non_empty)
    email = q.ask("Enter a valid email address that you own: ", q.non_empty)
    confirmed = cog_wrapper.sign_up_user(user_name, password, email)
    while not confirmed:
        print(f"User {user_name} requires confirmation. Check {email} for "
              f"a verification code.")
    confirmation_code = q.ask("Enter the confirmation code from the email: ")
```

```

if not confirmation_code:
    if q.ask("Do you need another confirmation code (y/n)? ", q.is_yesno):
        delivery = cog_wrapper.resend_confirmation(user_name)
        print(f"Confirmation code sent by {delivery['DeliveryMedium']} "
              f"to {delivery['Destination']}.")

else:
    confirmed = cog_wrapper.confirm_user_sign_up(user_name, confirmation_code)
print(f"User {user_name} is confirmed and ready to use.")
print('*'*88)

print("Let's get a list of users in the user pool.")
q.ask("Press Enter when you're ready.")
users = cog_wrapper.list_users()
if users:
    print(f"Found {len(users)} users:")
    pp(users)
else:
    print("No users found.")
print('*'*88)

print("Let's sign in and get an access token.")
auth_tokens = None
challenge = 'ADMIN_USER_PASSWORD_AUTH'
response = {}
while challenge is not None:
    if challenge == 'ADMIN_USER_PASSWORD_AUTH':
        response = cog_wrapper.start_sign_in(user_name, password)
        challenge = response['ChallengeName']
    elif response['ChallengeName'] == 'MFA_SETUP':
        print("First, we need to set up an MFA application.")
        qr_img = qrcode.make(
            f"otpauth://totp/{user_name}?secret={response['SecretCode']}")
        qr_img.save("qr.png")
        q.ask("Press Enter to see a QR code on your screen. Scan it into an MFA "
              "application, such as Google Authenticator.")
        webbrowser.open("qr.png")
        mfa_code = q.ask(
            "Enter the verification code from your MFA application: ",
            q.non_empty)
        response = cog_wrapper.verify_mfa(response['Session'], mfa_code)
        print(f"MFA device setup {response['Status']}")
        print("Now that an MFA application is set up, let's sign in again.")
        print("You might have to wait a few seconds for a new MFA code to appear in"
              "your MFA application.")
        challenge = 'ADMIN_USER_PASSWORD_AUTH'
    elif response['ChallengeName'] == 'SOFTWARE_TOKEN_MFA':
        auth_tokens = None
        while auth_tokens is None:
            mfa_code = q.ask(
                "Enter a verification code from your MFA application: ",
                q.non_empty)
            auth_tokens = cog_wrapper.respond_to_mfa_challenge(
                user_name, response['Session'], mfa_code)
        print(f"You're signed in as {user_name}.")
        print("Here's your access token:")
        pp(auth_tokens['AccessToken'])
        print("And your device information:")
        pp(auth_tokens['NewDeviceMetadata'])
        challenge = None
    else:
        raise Exception(f"Got unexpected challenge {response['ChallengeName']}")

print('*'*88)

device_group_key = auth_tokens['NewDeviceMetadata']['DeviceGroupKey']
device_key = auth_tokens['NewDeviceMetadata']['DeviceKey']
device_password = base64.standard_b64encode(os.urandom(40)).decode('utf-8')

```

```
    print("Let's confirm your MFA device so you don't have re-enter MFA tokens for it.")
    q.ask("Press Enter when you're ready.")
    cog_wrapper.confirm_mfa_device(
        user_name, device_key, device_group_key, device_password,
        auth_tokens['AccessToken'], aws_srp)
    print(f"Your device {device_key} is confirmed.")
    print('*'*88)

    print(f"Now let's sign in as {user_name} from your confirmed device {device_key}.")
    \n"
    f"Because this device is tracked by Amazon Cognito, you won't have to re-enter an MFA code.")
    q.ask("Press Enter when ready.")
    auth_tokens = cog_wrapper.sign_in_with_tracked_device(
        user_name, password, device_key, device_group_key, device_password, aws_srp)
    print("You're signed in. Your access token is:")
    pp(auth_tokens['AccessToken'])
    print('*'*88)

    print("Don't forget to delete your user pool when you're done with this example.")
    print("\nThanks for watching!")
    print('*'*88)

def main():
    parser = argparse.ArgumentParser(
        description="Shows how to sign up a new user with Amazon Cognito and associate "
                    "the user with an MFA application for multi-factor authentication.")
    parser.add_argument('user_pool_id', help="The ID of the user pool to use for the example.")
    parser.add_argument('client_id', help="The ID of the client application to use for the example.")
    args = parser.parse_args()
    try:
        run_scenario(boto3.client('cognito-idp'), args.user_pool_id, args.client_id)
    except Exception:
        logging.exception("Something went wrong with the demo.")

if __name__ == '__main__':
    main()
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [Admin GetUser](#)
  - [Admin Initiate Auth](#)
  - [Admin Respond To Auth Challenge](#)
  - [Associate Software Token](#)
  - [Confirm Device](#)
  - [Confirm Sign Up](#)
  - [Initiate Auth](#)
  - [List Users](#)
  - [Resend Confirmation Code](#)
  - [Respond To Auth Challenge](#)
  - [Sign Up](#)
  - [Verify Software Token](#)

## Amazon Comprehend examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Comprehend.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3829\)](#)
- [Scenarios \(p. 3840\)](#)

## Actions

### Create a document classifier

The following code example shows how to create an Amazon Comprehend document classifier.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendClassifier:  
    """Encapsulates an Amazon Comprehend custom classifier."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
        self.classifier_arn = None  
  
    def create(  
        self, name, language_code, training_bucket, training_key,  
        data_access_role_arn, mode):  
        """  
        Creates a custom classifier. After the classifier is created, it immediately  
        starts training on the data found in the specified Amazon S3 bucket. Training  
        can take 30 minutes or longer. The `describe_document_classifier` function  
        can be used to get training status and returns a status of TRAINED when the  
        classifier is ready to use.  
  
        :param name: The name of the classifier.  
        :param language_code: The language the classifier can operate on.  
        :param training_bucket: The Amazon S3 bucket that contains the training data.  
        :param training_key: The prefix used to find training data in the training  
            bucket. If multiple objects have the same prefix, all  
            of them are used.  
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that  
            grants Comprehend permission to read from the  
            training bucket.  
        :return: The ARN of the newly created classifier.  
        """  
        try:  
            response = self.comprehend_client.create_document_classifier(  
                DocumentClassifierName=name,
```

```
        LanguageCode=language_code,
        InputDataConfig={'S3Uri': f's3://{training_bucket}/{training_key}'},
        DataAccessRoleArn=data_access_role_arn,
        Mode=mode.value)
    self.classifier_arn = response['DocumentClassifierArn']
    logger.info("Started classifier creation. Arn is: %s.",
    self.classifier_arn)
except ClientError:
    logger.exception("Couldn't create classifier %s.", name)
    raise
else:
    return self.classifier_arn
```

- For API details, see [CreateDocumentClassifier in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete a document classifier

The following code example shows how to delete an Amazon Comprehend document classifier.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None

    def delete(self):
        """
        Deletes the classifier.
        """
        try:
            self.comprehend_client.delete_document_classifier(
                DocumentClassifierArn=self.classifier_arn)
            logger.info("Deleted classifier %s.", self.classifier_arn)
            self.classifier_arn = None
        except ClientError:
            logger.exception("Couldn't deleted classifier %s.", self.classifier_arn)
            raise
```

- For API details, see [DeleteDocumentClassifier in AWS SDK for Python \(Boto3\) API Reference](#).

## Describe a document classification job

The following code example shows how to describe an Amazon Comprehend document classification job.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendClassifier:  
    """Encapsulates an Amazon Comprehend custom classifier."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
        self.classifier_arn = None  
  
    def describe_job(self, job_id):  
        """  
        Gets metadata about a classification job.  
  
        :param job_id: The ID of the job to look up.  
        :return: Metadata about the job.  
        """  
        try:  
            response = self.comprehend_client.describe_document_classification_job(  
                JobId=job_id)  
            job = response['DocumentClassificationJobProperties']  
            logger.info("Got classification job %s.", job['JobName'])  
        except ClientError:  
            logger.exception("Couldn't get classification job %s.", job_id)  
            raise  
        else:  
            return job
```

- For API details, see [DescribeDocumentClassificationJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a document classifier

The following code example shows how to describe an Amazon Comprehend document classifier.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendClassifier:  
    """Encapsulates an Amazon Comprehend custom classifier."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
        self.classifier_arn = None  
  
    def describe(self, classifier_arn=None):  
        """  
        Gets metadata about a custom classifier, including its current status.  
  
        :param classifier_arn: The ARN of the classifier to look up.  
        :return: Metadata about the classifier.  
        """  
        if classifier_arn is not None:  
            self.classifier_arn = classifier_arn  
        try:  
            response = self.comprehend_client.describe_document_classifier(
```

```
        DocumentClassifierArn=self.classifier_arn)
    classifier = response['DocumentClassifierProperties']
    logger.info("Got classifier %s.", self.classifier_arn)
except ClientError:
    logger.exception("Couldn't get classifier %s.", self.classifier_arn)
    raise
else:
    return classifier
```

- For API details, see [DescribeDocumentClassifier](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a topic modeling job

The following code example shows how to describe an Amazon Comprehend topic modeling job.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client

    def describe_job(self, job_id):
        """
        Gets metadata about a topic modeling job.

        :param job_id: The ID of the job to look up.
        :return: Metadata about the job.
        """
        try:
            response = self.comprehend_client.describe_topics_detection_job(
                JobId=job_id)
            job = response['TopicsDetectionJobProperties']
            logger.info("Got topic detection job %s.", job_id)
        except ClientError:
            logger.exception("Couldn't get topic detection job %s.", job_id)
            raise
        else:
            return job
```

- For API details, see [DescribeTopicsDetectionJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect entities in a document

The following code example shows how to detect entities in a document with Amazon Comprehend.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendDetect:  
    """Encapsulates Comprehend detection functions."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
  
    def detect_entities(self, text, language_code):  
        """  
        Detects entities in a document. Entities can be things like people and places  
        or other common terms.  
  
        :param text: The document to inspect.  
        :param language_code: The language of the document.  
        :return: The list of entities along with their confidence scores.  
        """  
        try:  
            response = self.comprehend_client.detect_entities(  
                Text=text, LanguageCode=language_code)  
            entities = response['Entities']  
            logger.info("Detected %s entities.", len(entities))  
        except ClientError:  
            logger.exception("Couldn't detect entities.")  
            raise  
        else:  
            return entities
```

- For API details, see [DetectEntities](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect key phrases in a document

The following code example shows how to detect key phrases in a document with Amazon Comprehend.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendDetect:  
    """Encapsulates Comprehend detection functions."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
  
    def detect_key_phrases(self, text, language_code):  
        """  
        Detects key phrases in a document. A key phrase is typically a noun and its  
        modifiers.  
  
        :param text: The document to inspect.  
        :param language_code: The language of the document.  
        :return: The list of key phrases along with their confidence scores.  
        """  
        try:
```

```
        response = self.comprehend_client.detect_key_phrases(
            Text=text, LanguageCode=language_code)
        phrases = response['KeyPhrases']
        logger.info("Detected %s phrases.", len(phrases))
    except ClientError:
        logger.exception("Couldn't detect phrases.")
        raise
    else:
        return phrases
```

- For API details, see [DetectKeyPhrases](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect personally identifiable information in a document

The following code example shows how to detect personally identifiable information (PII) in a document with Amazon Comprehend.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client

    def detect_pii(self, text, language_code):
        """
        Detects personally identifiable information (PII) in a document. PII can be
        things like names, account numbers, or addresses.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of PII entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_pii_entities(
                Text=text, LanguageCode=language_code)
            entities = response['Entities']
            logger.info("Detected %s PII entities.", len(entities))
        except ClientError:
            logger.exception("Couldn't detect PII entities.")
            raise
        else:
            return entities
```

- For API details, see [DetectPiiEntities](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect syntactical elements of a document

The following code example shows how to detect syntactical elements of a document with Amazon Comprehend.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendDetect:  
    """Encapsulates Comprehend detection functions."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
  
    def detect_syntax(self, text, language_code):  
        """  
        Detects syntactical elements of a document. Syntax tokens are portions of  
        text along with their use as parts of speech, such as nouns, verbs, and  
        interjections.  
  
        :param text: The document to inspect.  
        :param language_code: The language of the document.  
        :return: The list of syntax tokens along with their confidence scores.  
        """  
        try:  
            response = self.comprehend_client.detect_syntax(  
                Text=text, LanguageCode=language_code)  
            tokens = response['SyntaxTokens']  
            logger.info("Detected %s syntax tokens.", len(tokens))  
        except ClientError:  
            logger.exception("Couldn't detect syntax.")  
            raise  
        else:  
            return tokens
```

- For API details, see [DetectSyntax](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect the dominant language in a document

The following code example shows how to detect the dominant language in a document with Amazon Comprehend.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendDetect:  
    """Encapsulates Comprehend detection functions."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
  
    def detect_languages(self, text):
```

```
"""
Detects languages used in a document.

:param text: The document to inspect.
:return: The list of languages along with their confidence scores.
"""
try:
    response = self.comprehend_client.detect_dominant_language(Text=text)
    languages = response['Languages']
    logger.info("Detected %s languages.", len(languages))
except ClientError:
    logger.exception("Couldn't detect languages.")
    raise
else:
    return languages
```

- For API details, see [DetectDominantLanguage in AWS SDK for Python \(Boto3\) API Reference](#).

## Detect the sentiment of a document

The following code example shows how to detect the sentiment of a document with Amazon Comprehend.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client

    def detect_sentiment(self, text, language_code):
        """
        Detects the overall sentiment expressed in a document. Sentiment can
        be positive, negative, neutral, or a mixture.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The sentiments along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_sentiment(
                Text=text, LanguageCode=language_code)
            logger.info("Detected primary sentiment %s.", response['Sentiment'])
        except ClientError:
            logger.exception("Couldn't detect sentiment.")
            raise
        else:
            return response
```

- For API details, see [DetectSentiment in AWS SDK for Python \(Boto3\) API Reference](#).

## List document classification jobs

The following code example shows how to list Amazon Comprehend document classification jobs.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None

    def list_jobs(self):
        """
        Lists the classification jobs for the current account.

        :return: The list of jobs.
        """
        try:
            response = self.comprehend_client.list_document_classification_jobs()
            jobs = response['DocumentClassificationJobPropertiesList']
            logger.info("Got %s document classification jobs.", len(jobs))
        except ClientError:
            logger.exception("Couldn't get document classification jobs.", )
            raise
        else:
            return jobs
```

- For API details, see [ListDocumentClassificationJobs](#) in *AWS SDK for Python (Boto3) API Reference*.

## List document classifiers

The following code example shows how to list Amazon Comprehend document classifiers.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None

    def list(self):
        """
        Lists custom classifiers for the current account.
```

```
:return: The list of classifiers.  
"""  
try:  
    response = self.comprehend_client.list_document_classifiers()  
    classifiers = response['DocumentClassifierPropertiesList']  
    logger.info("Got %s classifiers.", len(classifiers))  
except ClientError:  
    logger.exception("Couldn't get classifiers.", )  
    raise  
else:  
    return classifiers
```

- For API details, see [ListDocumentClassifiers](#) in *AWS SDK for Python (Boto3) API Reference*.

## List topic modeling jobs

The following code example shows how to list Amazon Comprehend topic modeling jobs.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendTopicModeler:  
    """Encapsulates a Comprehend topic modeler."""  
    def __init__(self, comprehend_client):  
        """  
        :param comprehend_client: A Boto3 Comprehend client.  
        """  
        self.comprehend_client = comprehend_client  
  
    def list_jobs(self):  
        """  
        Lists topic modeling jobs for the current account.  
        :return: The list of jobs.  
        """  
        try:  
            response = self.comprehend_client.list_topics_detection_jobs()  
            jobs = response['TopicsDetectionJobPropertiesList']  
            logger.info("Got %s topic detection jobs.", len(jobs))  
        except ClientError:  
            logger.exception("Couldn't get topic detection jobs.")  
            raise  
        else:  
            return jobs
```

- For API details, see [ListTopicsDetectionJobs](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start a document classification job

The following code example shows how to start an Amazon Comprehend document classification job.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None

    def start_job(
            self, job_name, input_bucket, input_key, input_format, output_bucket,
            output_key, data_access_role_arn):
        """
        Starts a classification job. The classifier must be trained or the job
        will fail. Input is read from the specified Amazon S3 input bucket and
        written to the specified output bucket. Output data is stored in a tar
        archive compressed in gzip format. The job runs asynchronously, so you can
        call `describe_document_classification_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: The Amazon S3 bucket that contains input data.
        :param input_key: The prefix used to find input data in the input
                          bucket. If multiple objects have the same prefix, all
                          of them are used.
        :param input_format: The format of the input data, either one document per
                           file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                    grants Comprehend permission to read from the
                                    input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_document_classification_job(
                DocumentClassifierArn=self.classifier_arn,
                JobName=job_name,
                InputDataConfig={
                    'S3Uri': f's3://{input_bucket}/{input_key}',
                    'InputFormat': input_format.value},
                OutputDataConfig={'S3Uri': f's3://{output_bucket}/{output_key}'},
                DataAccessRoleArn=data_access_role_arn)
            logger.info(
                "Document classification job %s is %s.", job_name,
                response['JobStatus'])
        except ClientError:
            logger.exception("Couldn't start classification job %s.", job_name)
            raise
        else:
            return response
```

- For API details, see [StartDocumentClassificationJob](#) in *AWS SDK for Python (Boto3) API Reference*.

### Start a topic modeling job

The following code example shows how to start an Amazon Comprehend topic modeling job.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client

    def start_job(
        self, job_name, input_bucket, input_key, input_format, output_bucket,
        output_key, data_access_role_arn):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
        input bucket and written to the specified output bucket. Output data is stored
        in a tar archive compressed in gzip format. The job runs asynchronously, so you
        can call `describe_topics_detection_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
            bucket. If multiple objects have the same prefix, all
            of them are used.
        :param input_format: The format of the input data, either one document per
            file or one document per line.
        :param output_bucket: The Amazon S3 bucket where output data is written.
        :param output_key: The prefix prepended to the output data.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
            grants Comprehend permission to read from the
            input bucket and write to the output bucket.
        :return: Information about the job, including the job ID.
        """
        try:
            response = self.comprehend_client.start_topics_detection_job(
                JobName=job_name,
                DataAccessRoleArn=data_access_role_arn,
                InputDataConfig={
                    'S3Uri': f's3://[{input_bucket}]/[{input_key}]',
                    'InputFormat': input_format.value},
                OutputDataConfig={'S3Uri': f's3://[{output_bucket}]/[{output_key}]'})
            logger.info("Started topic modeling job %s.", response['JobId'])
        except ClientError:
            logger.exception("Couldn't start topic modeling job.")
            raise
        else:
            return response
```

- For API details, see [StartTopicsDetectionJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Detect document elements

The following code example shows how to:

- Detect languages in a document.
- Detect entities in a document.
- Detect key phrases in a document.
- Detect personally identifiable information (PII) in a document.
- Detect the sentiment of a document.
- Detect syntax elements in a document.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps Amazon Comprehend actions.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class ComprehendDetect:
    """Encapsulates Comprehend detection functions."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client

    def detect_languages(self, text):
        """
        Detects languages used in a document.

        :param text: The document to inspect.
        :return: The list of languages along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_dominant_language(Text=text)
            languages = response['Languages']
            logger.info("Detected %s languages.", len(languages))
        except ClientError:
            logger.exception("Couldn't detect languages.")
            raise
        else:
            return languages

    def detect_entities(self, text, language_code):
        """
        Detects entities in a document. Entities can be things like people and places
        or other common terms.

        :param text: The document to inspect.
        :param language_code: The language of the document.
        :return: The list of entities along with their confidence scores.
        """
        try:
            response = self.comprehend_client.detect_entities(
                Text=text, LanguageCode=language_code)
            entities = response['Entities']
```

```
        logger.info("Detected %s entities.", len(entities))
    except ClientError:
        logger.exception("Couldn't detect entities.")
        raise
    else:
        return entities

def detect_key_phrases(self, text, language_code):
    """
    Detects key phrases in a document. A key phrase is typically a noun and its
    modifiers.

    :param text: The document to inspect.
    :param language_code: The language of the document.
    :return: The list of key phrases along with their confidence scores.
    """
    try:
        response = self.comprehend_client.detect_key_phrases(
            Text=text, LanguageCode=language_code)
        phrases = response['KeyPhrases']
        logger.info("Detected %s phrases.", len(phrases))
    except ClientError:
        logger.exception("Couldn't detect phrases.")
        raise
    else:
        return phrases

def detect_pii(self, text, language_code):
    """
    Detects personally identifiable information (PII) in a document. PII can be
    things like names, account numbers, or addresses.

    :param text: The document to inspect.
    :param language_code: The language of the document.
    :return: The list of PII entities along with their confidence scores.
    """
    try:
        response = self.comprehend_client.detect_pii_entities(
            Text=text, LanguageCode=language_code)
        entities = response['Entities']
        logger.info("Detected %s PII entities.", len(entities))
    except ClientError:
        logger.exception("Couldn't detect PII entities.")
        raise
    else:
        return entities

def detect_sentiment(self, text, language_code):
    """
    Detects the overall sentiment expressed in a document. Sentiment can
    be positive, negative, neutral, or a mixture.

    :param text: The document to inspect.
    :param language_code: The language of the document.
    :return: The sentiments along with their confidence scores.
    """
    try:
        response = self.comprehend_client.detect_sentiment(
            Text=text, LanguageCode=language_code)
        logger.info("Detected primary sentiment %s.", response['Sentiment'])
    except ClientError:
        logger.exception("Couldn't detect sentiment.")
        raise
    else:
        return response
```

```
def detect_syntax(self, text, language_code):
    """
    Detects syntactical elements of a document. Syntax tokens are portions of
    text along with their use as parts of speech, such as nouns, verbs, and
    interjections.

    :param text: The document to inspect.
    :param language_code: The language of the document.
    :return: The list of syntax tokens along with their confidence scores.
    """
    try:
        response = self.comprehend_client.detect_syntax(
            Text=text, LanguageCode=language_code)
        tokens = response['SyntaxTokens']
        logger.info("Detected %s syntax tokens.", len(tokens))
    except ClientError:
        logger.exception("Couldn't detect syntax.")
        raise
    else:
        return tokens
```

Call functions on the wrapper class to detect entities, phrases, and more in a document.

```
def usage_demo():
    print('*'*88)
    print("Welcome to the Amazon Comprehend detection demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    comp_detect = ComprehendDetect(boto3.client('comprehend'))
    with open('detect_sample.txt') as sample_file:
        sample_text = sample_file.read()

    demo_size = 3

    print("Sample text used for this demo:")
    print('*'*88)
    print(sample_text)
    print('*'*88)

    print("Detecting languages.")
    languages = comp_detect.detect_languages(sample_text)
    pprint(languages)
    lang_code = languages[0]['LanguageCode']

    print("Detecting entities.")
    entities = comp_detect.detect_entities(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(entities[:demo_size])

    print("Detecting key phrases.")
    phrases = comp_detect.detect_key_phrases(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(phrases[:demo_size])

    print("Detecting personally identifiable information (PII).")
    pii_entities = comp_detect.detect_pii(sample_text, lang_code)
    print(f"The first {demo_size} are:")
    pprint(pii_entities[:demo_size])

    print("Detecting sentiment.")
    sentiment = comp_detect.detect_sentiment(sample_text, lang_code)
    print(f"Sentiment: {sentiment['Sentiment']}")
```

```
print("SentimentScore:")
pprint(sentiment['SentimentScore'])

print("Detecting syntax elements.")
syntax_tokens = comp_detect.detect_syntax(sample_text, lang_code)
print(f"The first {demo_size} are:")
pprint(syntax_tokens[:demo_size])

print("Thanks for watching!")
print('-'*88)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [DetectDominantLanguage](#)
  - [DetectEntities](#)
  - [DetectKeyPhrases](#)
  - [DetectPiiEntities](#)
  - [DetectSentiment](#)
  - [DetectSyntax](#)

## Run a topic modeling job on sample data

The following code example shows how to:

- Run an Amazon Comprehend topic modeling job on sample data.
- Get information about the job.
- Extract job output data from Amazon Simple Storage Service (Amazon S3).

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a wrapper class to call Amazon Comprehend topic modeling actions.

```
class ComprehendTopicModeler:
    """Encapsulates a Comprehend topic modeler."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client

    def start_job(
            self, job_name, input_bucket, input_key, input_format, output_bucket,
            output_key, data_access_role_arn):
        """
        Starts a topic modeling job. Input is read from the specified Amazon S3
        input bucket and written to the specified output bucket. Output data is stored
        in a tar archive compressed in gzip format. The job runs asynchronously, so you
        can call `describe_topics_detection_job` to get job status until it
        returns a status of SUCCEEDED.

        :param job_name: The name of the job.
        :param input_bucket: An Amazon S3 bucket that contains job input.
        :param input_key: The prefix used to find input data in the input
                         bucket. If multiple objects have the same prefix, all
```

```

        of them are used.
:param input_format: The format of the input data, either one document per
        file or one document per line.
:param output_bucket: The Amazon S3 bucket where output data is written.
:param output_key: The prefix prepended to the output data.
:param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
        grants Comprehend permission to read from the
        input bucket and write to the output bucket.
:return: Information about the job, including the job ID.
"""
try:
    response = self.comprehend_client.start_topics_detection_job(
        JobName=job_name,
        DataAccessRoleArn=data_access_role_arn,
        InputDataConfig={
            'S3Uri': f's3://{input_bucket}/{input_key}',
            'InputFormat': input_format.value},
        OutputDataConfig={'S3Uri': f's3://{output_bucket}/{output_key}'})
    logger.info("Started topic modeling job %s.", response['JobId'])
except ClientError:
    logger.exception("Couldn't start topic modeling job.")
    raise
else:
    return response

def describe_job(self, job_id):
"""
Gets metadata about a topic modeling job.

:param job_id: The ID of the job to look up.
:return: Metadata about the job.
"""
try:
    response = self.comprehend_client.describe_topics_detection_job(
        JobId=job_id)
    job = response['TopicsDetectionJobProperties']
    logger.info("Got topic detection job %s.", job_id)
except ClientError:
    logger.exception("Couldn't get topic detection job %s.", job_id)
    raise
else:
    return job

def list_jobs(self):
"""
Lists topic modeling jobs for the current account.

:return: The list of jobs.
"""
try:
    response = self.comprehend_client.list_topics_detection_jobs()
    jobs = response['TopicsDetectionJobPropertiesList']
    logger.info("Got %s topic detection jobs.", len(jobs))
except ClientError:
    logger.exception("Couldn't get topic detection jobs.")
    raise
else:
    return jobs

```

Use the wrapper class to run a topic modeling job and get job data.

```

def usage_demo():
    print('*'*88)
    print("Welcome to the Amazon Comprehend topic modeling demo!")

```

```
print('*'*88)

logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

input_prefix = 'input/'
output_prefix = 'output/'
demo_resources = ComprehendDemoResources(
    boto3.resource('s3'), boto3.resource('iam'))
topic_modeler = ComprehendTopicModeler(boto3.client('comprehend'))

print("Setting up storage and security resources needed for the demo.")
demo_resources.setup('comprehend-topic-modeler-demo')
print("Copying sample data from public bucket into input bucket.")
demo_resources.bucket.copy(
    {'Bucket': 'public-sample-us-west-2', 'Key': 'TopicModeling/Sample.txt'},
    f'{input_prefix}sample.txt')

print("Starting topic modeling job on sample data.")
job_info = topic_modeler.start_job(
    'demo-topic-modeling-job', demo_resources.bucket.name, input_prefix,
    JobInputFormat.per_line, demo_resources.bucket.name, output_prefix,
    demo_resources.data_access_role.arn)

print(f"Waiting for job {job_info['JobId']} to complete. This typically takes "
      f"20 - 30 minutes.")
job_waiter = JobCompleteWaiter(topic_modeler.comprehend_client)
job_waiter.wait(job_info['JobId'])

job = topic_modeler.describe_job(job_info['JobId'])
print(f"Job {job['JobId']} complete:")
pprint(job)

print(f"Getting job output data from the output Amazon S3 bucket: "
      f"{job['OutputDataConfig']['S3Uri']}.")
job_output = demo_resources.extract_job_output(job)
lines = 10
print(f"First {lines} lines of document topics output:")
pprint(job_output['doc-topics.csv']['data'][:lines])
print(f"First {lines} lines of terms output:")
pprint(job_output['topic-terms.csv']['data'][:lines])

print("Cleaning up resources created for the demo.")
demo_resources.cleanup()

print("Thanks for watching!")
print('*'*88)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [DescribeTopicsDetectionJob](#)
  - [ListTopicsDetectionJobs](#)
  - [StartTopicsDetectionJob](#)

## Train a custom classifier and classify documents

The following code example shows how to:

- Create an Amazon Comprehend multi-label classifier.
- Train the classifier on sample data.
- Run a classification job on a second set of data.
- Extract the job output data from Amazon Simple Storage Service (Amazon S3).

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a wrapper class to call Amazon Comprehend document classifier actions.

```
class ComprehendClassifier:
    """Encapsulates an Amazon Comprehend custom classifier."""
    def __init__(self, comprehend_client):
        """
        :param comprehend_client: A Boto3 Comprehend client.
        """
        self.comprehend_client = comprehend_client
        self.classifier_arn = None

    def create(
        self, name, language_code, training_bucket, training_key,
        data_access_role_arn, mode):
        """
        Creates a custom classifier. After the classifier is created, it immediately
        starts training on the data found in the specified Amazon S3 bucket. Training
        can take 30 minutes or longer. The `describe_document_classifier` function
        can be used to get training status and returns a status of TRAINED when the
        classifier is ready to use.

        :param name: The name of the classifier.
        :param language_code: The language the classifier can operate on.
        :param training_bucket: The Amazon S3 bucket that contains the training data.
        :param training_key: The prefix used to find training data in the training
            bucket. If multiple objects have the same prefix, all
            of them are used.
        :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
            grants Comprehend permission to read from the
            training bucket.
        :return: The ARN of the newly created classifier.
        """
        try:
            response = self.comprehend_client.create_document_classifier(
                DocumentClassifierName=name,
                LanguageCode=language_code,
                InputDataConfig={'S3Uri': f's3://{training_bucket}/{training_key}'},
                DataAccessRoleArn=data_access_role_arn,
                Mode=mode.value)
            self.classifier_arn = response['DocumentClassifierArn']
            logger.info("Started classifier creation. Arn is: %s.",
                        self.classifier_arn)
        except ClientError:
            logger.exception("Couldn't create classifier %s.", name)
            raise
        else:
            return self.classifier_arn

    def describe(self, classifier_arn=None):
        """
        Gets metadata about a custom classifier, including its current status.

        :param classifier_arn: The ARN of the classifier to look up.
        :return: Metadata about the classifier.
        """
        if classifier_arn is not None:
            self.classifier_arn = classifier_arn
        try:
            response = self.comprehend_client.describe_document_classifier(
```

```
        DocumentClassifierArn=self.classifier_arn)
    classifier = response['DocumentClassifierProperties']
    logger.info("Got classifier %s.", self.classifier_arn)
except ClientError:
    logger.exception("Couldn't get classifier %s.", self.classifier_arn)
    raise
else:
    return classifier

def list(self):
    """
    Lists custom classifiers for the current account.

    :return: The list of classifiers.
    """
try:
    response = self.comprehend_client.list_document_classifiers()
    classifiers = response['DocumentClassifierPropertiesList']
    logger.info("Got %s classifiers.", len(classifiers))
except ClientError:
    logger.exception("Couldn't get classifiers.", )
    raise
else:
    return classifiers

def delete(self):
    """
    Deletes the classifier.

    """
try:
    self.comprehend_client.delete_document_classifier(
        DocumentClassifierArn=self.classifier_arn)
    logger.info("Deleted classifier %s.", self.classifier_arn)
    self.classifier_arn = None
except ClientError:
    logger.exception("Couldn't deleted classifier %s.", self.classifier_arn)
    raise

def start_job(
    self, job_name, input_bucket, input_key, input_format, output_bucket,
    output_key, data_access_role_arn):
    """
    Starts a classification job. The classifier must be trained or the job
    will fail. Input is read from the specified Amazon S3 input bucket and
    written to the specified output bucket. Output data is stored in a tar
    archive compressed in gzip format. The job runs asynchronously, so you can
    call `describe_document_classification_job` to get job status until it
    returns a status of SUCCEEDED.

    :param job_name: The name of the job.
    :param input_bucket: The Amazon S3 bucket that contains input data.
    :param input_key: The prefix used to find input data in the input
                      bucket. If multiple objects have the same prefix, all
                      of them are used.
    :param input_format: The format of the input data, either one document per
                        file or one document per line.
    :param output_bucket: The Amazon S3 bucket where output data is written.
    :param output_key: The prefix prepended to the output data.
    :param data_access_role_arn: The Amazon Resource Name (ARN) of a role that
                                grants Comprehend permission to read from the
                                input bucket and write to the output bucket.
    :return: Information about the job, including the job ID.
    """
try:
    response = self.comprehend_client.start_document_classification_job(
        DocumentClassifierArn=self.classifier_arn,
```

```
JobName=job_name,
InputDataConfig={
    'S3Uri': f's3://{input_bucket}/{input_key}',
    'InputFormat': input_format.value},
OutputDataConfig={'S3Uri': f's3://{output_bucket}/{output_key}'},
DataAccessRoleArn=data_access_role_arn)
logger.info(
    "Document classification job %s is %s.", job_name,
    response['JobStatus'])
except ClientError:
    logger.exception("Couldn't start classification job %s.", job_name)
    raise
else:
    return response

def describe_job(self, job_id):
    """
    Gets metadata about a classification job.

    :param job_id: The ID of the job to look up.
    :return: Metadata about the job.
    """
    try:
        response = self.comprehend_client.describe_document_classification_job(
            JobId=job_id)
        job = response['DocumentClassificationJobProperties']
        logger.info("Got classification job %s.", job['JobName'])
    except ClientError:
        logger.exception("Couldn't get classification job %s.", job_id)
        raise
    else:
        return job

def list_jobs(self):
    """
    Lists the classification jobs for the current account.

    :return: The list of jobs.
    """
    try:
        response = self.comprehend_client.list_document_classification_jobs()
        jobs = response['DocumentClassificationJobPropertiesList']
        logger.info("Got %s document classification jobs.", len(jobs))
    except ClientError:
        logger.exception("Couldn't get document classification jobs.", )
        raise
    else:
        return jobs
```

Create a class to help run the scenario.

```
class ClassifierDemo:
    """
    Encapsulates functions used to run the demonstration.
    """
    def __init__(self, demo_resources):
        """
        :param demo_resources: A ComprehendDemoResources class that manages resources
                               for the demonstration.
        """
        self.demo_resources = demo_resources
        self.training_prefix = 'training/'
        self.input_prefix = 'input/'
        self.input_format = JobInputFormat.per_line
```

```

        self.output_prefix = 'output/'

    def setup(self):
        """Creates AWS resources used by the demo."""
        self.demo_resources.setup('comprehend-classifier-demo')

    def cleanup(self):
        """Deletes AWS resources used by the demo."""
        self.demo_resources.cleanup()

    @staticmethod
    def _sanitize_text(text):
        """Removes characters that cause errors for the document parser."""
        return text.replace('\r', ' ').replace('\n', ' ').replace(',', ';')

    @staticmethod
    def _get_issues(query, issue_count):
        """
        Gets issues from GitHub using the specified query parameters.

        :param query: The query string used to request issues from the GitHub API.
        :param issue_count: The number of issues to retrieve.
        :return: The list of issues retrieved from GitHub.
        """

        issues = []
        logger.info("Requesting issues from %s?%s.", GITHUB_SEARCH_URL, query)
        response = requests.get(
            f'{GITHUB_SEARCH_URL}?{query}&per_page={issue_count}')
        if response.status_code == 200:
            issue_page = response.json()['items']
            logger.info("Got %s issues.", len(issue_page))
            issues = [
                {
                    'title': ClassifierDemo._sanitize_text(issue['title']),
                    'body': ClassifierDemo._sanitize_text(issue['body']),
                    'labels': {label['name'] for label in issue['labels']}
                } for issue in issue_page]
        else:
            logger.error(
                "GitHub returned error code %s with message %s.",
                response.status_code, response.json())
        logger.info("Found %s issues.", len(issues))
        return issues

    def get_training_issues(self, training_labels):
        """
        Gets issues used for training the custom classifier. Training issues are
        closed issues from the Boto3 repo that have known labels. Comprehend
        requires a minimum of ten training issues per label.

        :param training_labels: The issue labels to use for training.
        :return: The set of issues used for training.
        """

        issues = []
        per_label_count = 15
        for label in training_labels:
            issues += self._get_issues(
                f'q=type:issue+repo:boto/boto3+state:closed+label:{label}',
                per_label_count)
            for issue in issues:
                issue['labels'] = issue['labels'].intersection(training_labels)
        return issues

    def get_input_issues(self, training_labels):
        """
        Gets input issues from GitHub. For demonstration purposes, input issues
        are open issues from the Boto3 repo with known labels, though in practice

```

```

any issue could be submitted to the classifier for labeling.

:param training_labels: The set of labels to query for.
:return: The set of issues used for input.
"""
issues = []
per_label_count = 5
for label in training_labels:
    issues += self._get_issues(
        f'q=type:issue+repo:boto/boto3+state:open+label:{label}',
        per_label_count)
return issues

def upload_issue_data(self, issues, training=False):
"""
Uploads issue data to an Amazon S3 bucket, either for training or for input.
The data is first put into the format expected by Comprehend. For training,
the set of pipe-delimited labels is prepended to each document. For
input, labels are not sent.

:param issues: The set of issues to upload to Amazon S3.
:param training: Indicates whether the issue data is used for training or
                 input.
"""
try:
    obj_key = (
        self.training_prefix if training else self.input_prefix) + 'issues.txt'
    if training:
        issue_strings = [
            f"{'|'.join(issue['labels'])},{issue['title']} {issue['body']}"
            for issue in issues]
    else:
        issue_strings = [
            f"{issue['title']} {issue['body']}" for issue in issues]
    issue_bytes = BytesIO('\n'.join(issue_strings).encode('utf-8'))
    self.demo_resources.bucket.upload_fileobj(issue_bytes, obj_key)
    logger.info(
        "Uploaded data as %s to bucket %s.", obj_key,
        self.demo_resources.bucket.name)
except ClientError:
    logger.exception(
        "Couldn't upload data to bucket %s.", self.demo_resources.bucket.name)
    raise

def extract_job_output(self, job):
"""
Extracts job output from Amazon S3.
"""
return self.demo_resources.extract_job_output(job)

@staticmethod
def reconcile_job_output(input_issues, output_dict):
"""
Reconciles job output with the list of input issues. Because the input issues
have known labels, these can be compared with the labels added by the
classifier to judge the accuracy of the output.

:param input_issues: The list of issues used as input.
:param output_dict: The dictionary of data that is output by the classifier.
:return: The list of reconciled input and output data.
"""
reconciled = []
for archive in output_dict.values():
    for line in archive['data']:
        in_line = int(line['Line'])
        in_labels = input_issues[in_line]['labels']
        out_labels = {label['Name'] for label in line['Labels']
                     if float(label['Score']) > 0.3}

```

```
reconciled.append(
    f"\"{line['File']}\", line {in_line} has labels {in_labels}.\\n"
    f"\\tClassifier assigned {out_labels}.\")")
logger.info("Reconciled input and output labels.")
return reconciled
```

Train a classifier on a set of GitHub issues with known labels, then send a second set of GitHub issues to the classifier so that they can be labeled.

```
def usage_demo():
    print('-'*88)
    print("Welcome to the Amazon Comprehend custom document classifier demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    comp_demo = ClassifierDemo(ComprehendDemoResources(
        boto3.resource('s3'), boto3.resource('iam')))
    comp_classifier = ComprehendClassifier(boto3.client('comprehend'))
    classifier_trained_waiter = ClassifierTrainedWaiter(
        comp_classifier.comprehend_client)
    training_labels = {'bug', 'feature-request', 'dynamodb', 's3'}

    print("Setting up storage and security resources needed for the demo.")
    comp_demo.setup()

    print("Getting training data from GitHub and uploading it to Amazon S3.")
    training_issues = comp_demo.get_training_issues(training_labels)
    comp_demo.upload_issue_data(training_issues, True)

    classifier_name = 'doc-example-classifier'
    print(f"Creating document classifier {classifier_name}.")
    comp_classifier.create(
        classifier_name, 'en',
        comp_demo.demo_resources.bucket.name,
        comp_demo.training_prefix,
        comp_demo.demo_resources.data_access_role.arn,
        ClassifierMode.multi_label)
    print(f"Waiting until {classifier_name} is trained. This typically takes "
          f"30-40 minutes.")
    classifier_trained_waiter.wait(comp_classifier.classifier_arn)

    print(f"Classifier {classifier_name} is trained:")
    pprint(comp_classifier.describe())

    print("Getting input data from GitHub and uploading it to Amazon S3.")
    input_issues = comp_demo.get_input_issues(training_labels)
    comp_demo.upload_issue_data(input_issues)

    print("Starting classification job on input data.")
    job_info = comp_classifier.start_job(
        'issue_classification_job',
        comp_demo.demo_resources.bucket.name,
        comp_demo.input_prefix,
        comp_demo.input_format,
        comp_demo.demo_resources.bucket.name,
        comp_demo.output_prefix,
        comp_demo.demo_resources.data_access_role.arn)
    print(f"Waiting for job {job_info['JobId']} to complete.")
    job_waiter = JobCompleteWaiter(comp_classifier.comprehend_client)
    job_waiter.wait(job_info['JobId'])

    job = comp_classifier.describe_job(job_info['JobId'])
```

```
print(f"Job {job['JobId']} complete:")
pprint(job)

print(f"Getting job output data from Amazon S3: "
      f"{job['OutputDataConfig']['S3Uri']}.")
job_output = comp_demo.extract_job_output(job)
print("Job output:")
pprint(job_output)

print("Reconciling job output with labels from GitHub:")
reconciled_output = comp_demo.reconcile_job_output(input_issues, job_output)
print(*reconciled_output, sep='\n')

answer = input(f"Do you want to delete the classifier {classifier_name} (y/n)? ")
if answer.lower() == 'y':
    print(f"Deleting {classifier_name}.")
    comp_classifier.delete()

print("Cleaning up resources created for the demo.")
comp_demo.cleanup()

print("Thanks for watching!")
print('-'*88)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateDocumentClassifier](#)
  - [DeleteDocumentClassifier](#)
  - [DescribeDocumentClassificationJob](#)
  - [DescribeDocumentClassifier](#)
  - [ListDocumentClassificationJobs](#)
  - [ListDocumentClassifiers](#)
  - [StartDocumentClassificationJob](#)

## AWS Config examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with AWS Config.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3853\)](#)

## Actions

### Delete a rule

The following code example shows how to delete an AWS Config rule.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ConfigWrapper:  
    """  
    Encapsulates AWS Config functions.  
    """  
    def __init__(self, config_client):  
        """  
        :param config_client: A Boto3 AWS Config client.  
        """  
        self.config_client = config_client  
  
    def delete_config_rule(self, rule_name):  
        """  
        Delete the specified rule.  
  
        :param rule_name: The name of the rule to delete.  
        """  
        try:  
            self.config_client.delete_config_rule(ConfigRuleName=rule_name)  
            logger.info("Deleted rule %s.", rule_name)  
        except ClientError:  
            logger.exception("Couldn't delete rule %s.", rule_name)  
            raise
```

- For API details, see [DeleteConfigRule](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe rules

The following code example shows how to describe AWS Config rules.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ConfigWrapper:  
    """  
    Encapsulates AWS Config functions.  
    """  
    def __init__(self, config_client):  
        """  
        :param config_client: A Boto3 AWS Config client.  
        """  
        self.config_client = config_client  
  
    def describe_config_rule(self, rule_name):  
        """  
        Gets data for the specified rule.  
  
        :param rule_name: The name of the rule to retrieve.  
        :return: The rule data.  
        """  
        try:  
            response = self.config_client.describe_config_rules(  
                ConfigRuleNames=[rule_name])  
            rule = response['ConfigRules'][0]  
            logger.info("Got data for rule %s.", rule_name)  
        except ClientError:  
            logger.exception("Couldn't get data for rule %s.", rule_name)
```

```
        raise
    else:
        return rule
```

- For API details, see [DescribeConfigRules](#) in *AWS SDK for Python (Boto3) API Reference*.

## Put a rule

The following code example shows how to put an AWS Config rule.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ConfigWrapper:
    """
    Encapsulates AWS Config functions.
    """
    def __init__(self, config_client):
        """
        :param config_client: A Boto3 AWS Config client.
        """
        self.config_client = config_client

    def put_config_rule(self, rule_name):
        """
        Sets a configuration rule that prohibits making Amazon S3 buckets publicly
        readable.

        :param rule_name: The name to give the rule.
        """
        try:
            self.config_client.put_config_rule(
                ConfigRule={
                    'ConfigRuleName': rule_name,
                    'Description': 'S3 Public Read Prohibited Bucket Rule',
                    'Scope': {
                        'ComplianceResourceTypes': [
                            'AWS::S3::Bucket',
                        ],
                    },
                    'Source': {
                        'Owner': 'AWS',
                        'SourceIdentifier': 'S3_BUCKET_PUBLIC_READ_PROHIBITED',
                    },
                    'InputParameters': '{}',
                    'ConfigRuleState': 'ACTIVE'
                }
            )
            logger.info("Created configuration rule %s.", rule_name)
        except ClientError:
            logger.exception("Couldn't create configuration rule %s.", rule_name)
            raise
```

- For API details, see [PutConfigRule](#) in *AWS SDK for Python (Boto3) API Reference*.

## Device Farm examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with AWS Device Farm.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Scenarios \(p. 3856\)](#)

## Scenarios

### Run browser tests and take screenshots

The following code example shows how to run browser tests with Device Farm and take screenshots.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Use PyTest and Selenium to browse to specified websites, take screenshots, and compare actual website content with expected content.

```
import datetime
import os
import subprocess
import boto3
import pytest
from selenium import webdriver
from selenium.webdriver import DesiredCapabilities
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait

def get_git_hash():
    """
    Get the short Git hash of the current commit of the repository
    """
    try:
        return subprocess.check_output(
            ['git', 'rev-parse', '--short', 'HEAD']).decode('utf-8').strip()
    except:
        return "norepo"

class TestHelloSuite:
    """
    Our test suite.

    This style of test suite allows us to use setup_method and teardown_method.
    """
    def save_screenshot(self, name):
        self.driver.save_screenshot(os.path.join(self.screenshot_path, name))
```

```
def setup_method(self, method):
    """
    Set up a test.

    This makes sure that the session for an individual test is ready.

    The AWS credentials are read from the default ~/.aws/credentials or from the
    command line by setting the AWS_ACCESS_KEY_ID and AWS_SECRET_KEY environment
    variables.

    The project Amazon Resource Name (ARN) is determined by the PROJECT_ARN
    environment variable.
    """
    devicefarm_client = boto3.client("devicefarm")
    project_arn = os.environ.get("PROJECT_ARN", None)
    if project_arn is None:
        raise ValueError("Must set PROJECT_ARN")
    # Request a driver hub URL for the Selenium client
    testgrid_url_response = devicefarm_client.create_test_grid_url(
        projectArn=project_arn,
        expiresInSeconds=300)

    # We want a directory to save our files into. We're going to make a directory
    # in the current directory that holds our results.
    self.screenshot_path = os.path.join(
        '.', 'results', get_git_hash() + '-' + (datetime.date.today().isoformat()))
    if not os.path.exists(self.screenshot_path):
        os.makedirs(self.screenshot_path, exist_ok=True)

    # We want a Firefox instance on Windows
    desired_cap = DesiredCapabilities.FIREFOX
    desired_cap['platform'] = 'windows'
    desired_cap['BrowserVersion'] = 'latest'

    # Configure the webdriver with the appropriate remote endpoint.
    self.driver = webdriver.Remote(testgrid_url_response['url'], desired_cap)

    #
    # Auto-Tagging
    #

    # In order to get the Session ARN, we need to look up the session by the
    # Project ARN and session ID (from the driver).
    testgrid_session_arn_response = devicefarm_client.get_test_grid_session(
        projectArn=project_arn,
        sessionId=self.driver.session_id
    )

    # Save the session's ARN so we can tag the session.
    self.session_arn = testgrid_session_arn_response['testGridSession']['arn']

    # In order to tag it, we're going to use the resourcegroupstaggingapi client to
    # add a tag to the session ARN that we just got.
    tag_client = boto3.client('resourcegroupstaggingapi')
    tag_client.tag_resources(
        ResourceARNList=[self.session_arn],
        Tags={
            "TestSuite": f"testsuite {method.__name__}",
            "GitId": get_git_hash()
        }
    )

def teardown_method(self, method):
    """
    Clean up resources used by each method.
    """
    # End the Selenium session so we're off the clock.
```

```

        self.driver.quit()

    @pytest.mark.parametrize(
        'query,leading', [
            pytest.param('Seattle', 'Seattle (/si#ætəl/ (listen) see-AT-əl) is a
seaport city on the West Coast of the United States.'),
            pytest.param('Selenium', "Selenium is a chemical element with the symbol Se
and atomic number 34."),
            pytest.param('Amazon Locker', 'Amazon Locker is a self-service package
delivery service offered by online retailer Amazon.'),
            pytest.param('Kootenai Falls', 'Kootenai Falls is a waterfall on the
Kootenay River located in Lincoln County, Montana, just off U.S. Route 2.'),
            pytest.param('Dorayaki', 'Dorayaki (####, ####, ####, ####) is a type of
Japanese confection.'),
            pytest.param('Robot Face', '<|°_°|> (also known as Robot Face or Robot)')
        ])
    def test_first_paragraph_text(self, query, leading):
        """
        This test looks at the first paragraph of a page on Wikipedia, comparing it to
        a known leading sentence.

        If the leading sentence matches, the test passes. A screenshot is taken before
        the final assertion is made, letting us debug if something isn't right.
        """
        # Open the main page of Wikipedia
        self.driver.get("https://en.wikipedia.org/wiki/Main_Page")
        # Find the search box, enter a query, and press enter
        search_input = self.driver.find_element(By.ID, "searchInput")
        search_input.click()
        search_input.send_keys(query)
        search_input.send_keys(Keys.ENTER)
        # Wait for the search box to go stale -- This means we've navigated fully.
        WebDriverWait(self.driver,
        5).until(expected_conditions.staleness_of(search_input))
        # Get the leading paragraph of the article.
        lead = leading.lower()
        # Find the element...
        lead_para = self.driver.find_element(
            By.XPATH, "//div[@class='mw-parser-output']//p[not(@class)]")
        # ... and copy out its text.
        our_text = lead_para.text.lower()
        our_text = our_text[:len(lead)]
        # Take a screenshot and compare the strings.
        self.save_screenshot(f"leadingpara_{query}.png")
        assert our_text.startswith(lead)

    @pytest.mark.parametrize(
        "query,expected", [
            pytest.param("Automation Testing", "Test Automation"),
            pytest.param("DevOps", "DevOps"),
            pytest.param("Jackdaws Love My Big Sphinx Of Quartz", "Pangram"),
            pytest.param("EarthBound", "EarthBound"),
            pytest.param("Covered Bridges Today", "Covered Bridges Today"),
            pytest.param("Kurt Godel", "Kurt Gödel"),
            pytest.param("N//ng language", "N#ng language"),
            pytest.param("Who the Fuck Is Jackson Pollock?", "Who the $&% Is Jackson
Pollock?")
        ])
    def test_redirect_titles(self, query, expected):
        """
        A test comparing pages we expect to (or not to) redirect on Wikipedia.

        This test checks to see that the page ("query") redirects (or doesn't) to the
        "expected" page title. Several of these are common synonyms ("Jackdaws...") while
        others are because of characters untypable by most keyboards ("N#ng language")
        """

```

```
A screenshot is taken just before the final assertion is made to aid in
debugging and verification.
"""
# Open the main page of Wikipedia
self.driver.get("https://en.wikipedia.org/wiki/Main_Page")
# Find the search box, enter some text into it, and send an enter key.
search_input = self.driver.find_element(By.ID, "searchInput")
search_input.click()
search_input.send_keys(query)
search_input.send_keys(Keys.ENTER)
# wait until the page has rolled over -- once the search input handle is stale,
# the browser has navigated.
WebDriverWait(self.driver,
5).until(expected_conditions.staleness_of(search_input))
# Get the first heading & take a screenshot
our_text = self.driver.find_element(By.ID, "firstHeading").text.lower()
self.save_screenshot(f"redirect_{query}.png")
# did it match?
assert our_text == expected.lower()
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateTestGridUrl](#)
  - [GetTestGridSession](#)

## Upload and test mobile device packages

The following code example shows how to upload and test mobile device packages with Device Farm.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Upload compiled Android application and test packages to Device Farm, start a test, wait for test completion, and report the results.

```
import boto3
import os
import requests
import string
import random
import datetime
import time

# Update this dict with your own values before you run the example:
config = {
    # This is our app under test.
    "appFilePath": "app-debug.apk",
    "projectArn": "arn:aws:devicefarm:us-west-2:111222333444:project:581f5703-
e040-4ac9-b7ae-0ba007bfb8e6",
    # Since we care about the most popular devices, we'll use a curated pool.
    "testSpecArn": "arn:aws:devicefarm:us-west-2::upload:20fcf771-eae3-4137-
aa76-92e17fb3131b",
    "poolArn": "arn:aws:devicefarm:us-
west-2::devicepool:4a869d91-6f17-491f-9a95-0a601aee2406",
    "namePrefix": "MyAppTest",
    # This is our test package. This tutorial won't go into how to make these.
    "testPackage": "tests.zip"
}
```

```

client = boto3.client('devicefarm')

unique =
config['namePrefix']+ "-" +(datetime.date.today().isoformat())+''.join(random.sample(string.ascii_l
8))

print(f"The unique identifier for this run is '{unique}'. All uploads will be prefixed
"
      f"with this.")

def upload_df_file(filename, type_, mime='application/octet-stream'):
    upload_response = client.create_upload(
        projectArn=config['projectArn'],
        name=unique+"_"+os.path.basename(filename),
        type=type_,
        contentType=mime)
    upload_arn = upload_response['upload']['arn']
    # Extract the URL of the upload and use Requests to upload it.
    upload_url = upload_response['upload']['url']
    with open(filename, 'rb') as file_stream:
        print(f"Uploading {filename} to Device Farm as "
              f"{upload_response['upload']['name']}... ", end='')
        put_req = requests.put(
            upload_url, data=file_stream, headers={"content-type": mime})
        print(' done')
        if not put_req.ok:
            raise Exception(f"Couldn't upload. Requests says: {put_req.reason}")
    started = datetime.datetime.now()
    while True:
        print(f"Upload of {filename} in state {upload_response['upload']['status']} "
              f"after "+str(datetime.datetime.now() - started))
        if upload_response['upload']['status'] == 'FAILED':
            raise Exception(
                f"The upload failed processing. Device Farm says the reason is: \n"
                f"{+upload_response['upload']['message']}")
        if upload_response['upload']['status'] == 'SUCCEEDED':
            break
        time.sleep(5)
        upload_response = client.get_upload(arn=upload_arn)
    print("")
    return upload_arn

our_upload_arn = upload_df_file(config['appFilePath'], "ANDROID_APP")
our_test_package_arn = upload_df_file(config['testPackage'],
    'APPIUM_PYTHON_TEST_PACKAGE')
print(our_upload_arn, our_test_package_arn)

response = client.schedule_run(
    projectArn=config["projectArn"],
    appArn=our_upload_arn,
    devicePoolArn=config["poolArn"],
    name=unique,
    test={
        "type" :"APPIUM_PYTHON",
        "testSpecArn": config["testSpecArn"],
        "testPackageArn": our_test_package_arn})
run_arn = response['run']['arn']
start_time = datetime.datetime.now()
print(f"Run {unique} is scheduled as arn {run_arn} ")

state = 'UNKNOWN'
try:
    while True:

```

```
response = client.get_run(arn=run_arn)
state = response['run']['status']
if state == 'COMPLETED' or state == 'ERRORED':
    break
else:
    print(f" Run {unique} in state {state}, total "
          f"time {datetime.datetime.now() - start_time}")
    time.sleep(10)
except:
    client.stop_run(arn=run_arn)
    exit(1)

print(f"Tests finished in state {state} after {datetime.datetime.now() - start_time}")
# Pull all the logs.
jobs_response = client.list_jobs(arn=run_arn)
# Save the output somewhere, using the unique value.
save_path = os.path.join(os.getcwd(), 'results', unique)
os.mkdir(save_path)
# Save the last run information.
for job in jobs_response['jobs']:
    job_name = job['name']
    os.makedirs(os.path.join(save_path, job_name), exist_ok=True)
    # Get each suite within the job.
    suites = client.list_suites(arn=job['arn'])['suites']
    for suite in suites:
        for test in client.list_tests(arn=suite['arn'])['tests']:
            # Get the artifacts.
            for artifact_type in ['FILE', 'SCREENSHOT', 'LOG']:
                artifacts = client.list_artifacts(
                    type=artifact_type, arn=test['arn'])['artifacts']
                for artifact in artifacts:
                    # Replace `:` because it has a special meaning in Windows & macOS.
                    path_to = os.path.join(
                        save_path, job_name, suite['name'], test['name'].replace(':', '_'))
                    os.makedirs(path_to, exist_ok=True)
                    filename =
artifact['type']+ "_" + artifact['name'] + "_" + artifact['extension']
                    artifact_save_path = os.path.join(path_to, filename)
                    print(f"Downloading {artifact_save_path}")
                    with open(artifact_save_path, 'wb') as fn:
                        with requests.get(artifact['url'], allow_redirects=True) as
request:
                            fn.write(request.content)
print("Finished")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateUpload](#)
  - [GetRun](#)
  - [GetUpload](#)
  - [ListArtifacts](#)
  - [ListJobs](#)
  - [ListSuites](#)
  - [ListTests](#)
  - [ScheduleRun](#)
  - [StopRun](#)

## DynamoDB examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon DynamoDB.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3862\)](#)
- [Scenarios \(p. 3875\)](#)

## Actions

### Create a table

The following code example shows how to create a DynamoDB table.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a table for storing movie data.

```
class Movies:  
    """Encapsulates an Amazon DynamoDB table of movie data."""  
    def __init__(self, dyn_resource):  
        """  
        :param dyn_resource: A Boto3 DynamoDB resource.  
        """  
        self.dyn_resource = dyn_resource  
        self.table = None  
  
    def create_table(self, table_name):  
        """  
        Creates an Amazon DynamoDB table that can be used to store movie data.  
        The table uses the release year of the movie as the partition key and the  
        title as the sort key.  
  
        :param table_name: The name of the table to create.  
        :return: The newly created table.  
        """  
        try:  
            self.table = self.dyn_resource.create_table(  
                TableName=table_name,  
                KeySchema=[  
                    {'AttributeName': 'year', 'KeyType': 'HASH'}, # Partition key  
                    {'AttributeName': 'title', 'KeyType': 'RANGE'} # Sort key  
                ],  
                AttributeDefinitions=[  
                    {'AttributeName': 'year', 'AttributeType': 'N'},  
                    {'AttributeName': 'title', 'AttributeType': 'S'}  
                ],  
                ProvisionedThroughput={'ReadCapacityUnits': 10, 'WriteCapacityUnits':  
10})  
            self.table.wait_until_exists()  
        except ClientError as err:  
            logger.error(
```

```
        "Couldn't create table %s. Here's why: %s: %s", table_name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return self.table
```

- For API details, see [CreateTable](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a table

The following code example shows how to delete a DynamoDB table.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None

    def delete_table(self):
        """
        Deletes the table.
        """
        try:
            self.table.delete()
            self.table = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

- For API details, see [DeleteTable](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an item from a table

The following code example shows how to delete an item from a DynamoDB table.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
```

```
    self.dyn_resource = dyn_resource
    self.table = None

    def delete_movie(self, title, year):
        """
        Deletes a movie from the table.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        """
        try:
            self.table.delete_item(Key={'year': year, 'title': title})
        except ClientError as err:
            logger.error(
                "Couldn't delete movie %s. Here's why: %s: %s",
                title,
                err.response['Error']['Code'],
                err.response['Error']['Message']
            )
            raise
```

You can specify a condition so that an item is deleted only when it meets certain criteria.

```
class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table

    def delete_underrated_movie(self, title, year, rating):
        """
        Deletes a movie only if it is rated below a specified value. By using a
        condition expression in a delete operation, you can specify that an item is
        deleted only when it meets certain criteria.

        :param title: The title of the movie to delete.
        :param year: The release year of the movie to delete.
        :param rating: The rating threshold to check before deleting the movie.
        """
        try:
            self.table.delete_item(
                Key={'year': year, 'title': title},
                ConditionExpression="info.rating <= :val",
                ExpressionAttributeValues={":val": Decimal(str(rating))})
        except ClientError as err:
            if err.response['Error']['Code'] == "ConditionalCheckFailedException":
                logger.warning(
                    "Didn't delete %s because its rating is greater than %s.",
                    title, rating)
            else:
                logger.error(
                    "Couldn't delete movie %s. Here's why: %s: %s",
                    title,
                    err.response['Error']['Code'],
                    err.response['Error']['Message'])
            raise
```

- For API details, see [DeleteItem](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a batch of items

The following code example shows how to get a batch of DynamoDB items.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import decimal
import json
import logging
import os
import pprint
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
dynamodb = boto3.resource('dynamodb')

MAX_GET_SIZE = 100 # Amazon DynamoDB rejects a get batch larger than 100 items.

def do_batch_get(batch_keys):
    """
    Gets a batch of items from Amazon DynamoDB. Batches can contain keys from
    more than one table.

    When Amazon DynamoDB cannot process all items in a batch, a set of unprocessed
    keys is returned. This function uses an exponential backoff algorithm to retry
    getting the unprocessed keys until all are retrieved or the specified
    number of tries is reached.

    :param batch_keys: The set of keys to retrieve. A batch can contain at most 100
                       keys. Otherwise, Amazon DynamoDB returns an error.
    :return: The dictionary of retrieved items grouped under their respective
             table names.
    """
    tries = 0
    max_tries = 5
    sleepy_time = 1 # Start with 1 second of sleep, then exponentially increase.
    retrieved = {key: [] for key in batch_keys}
    while tries < max_tries:
        response = dynamodb.batch_get_item(RequestItems=batch_keys)
        # Collect any retrieved items and retry unprocessed keys.
        for key in response.get('Responses', []):
            retrieved[key] += response['Responses'][key]
        unprocessed = response['UnprocessedKeys']
        if len(unprocessed) > 0:
            batch_keys = unprocessed
            unprocessed_count = sum(
                [len(batch_key['Keys']) for batch_key in batch_keys.values()])
            logger.info(
                "%s unprocessed keys returned. Sleep, then retry.", unprocessed_count)
            tries += 1
            if tries < max_tries:
                logger.info("Sleeping for %s seconds.", sleepy_time)
                time.sleep(sleepy_time)
                sleepy_time = min(sleepy_time * 2, 32)
            else:
                break
    return retrieved
```

- For API details, see [BatchGetItem](#) in [AWS SDK for Python \(Boto3\) API Reference](#).

## Get an item from a table

The following code example shows how to get an item from a DynamoDB table.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None

    def get_movie(self, title, year):
        """
        Gets movie data from the table for a specific movie.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :return: The data about the requested movie.
        """
        try:
            response = self.table.get_item(Key={'year': year, 'title': title})
        except ClientError as err:
            logger.error(
                "Couldn't get movie %s from table %s. Here's why: %s: %s",
                title, self.table.name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['Item']
```

- For API details, see [GetItem](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get information about a table

The following code example shows how to get information about a DynamoDB table.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None

    def exists(self, table_name):
        """
        Determines whether a table exists. As a side effect, stores the table in
        a member variable.
```

```
:param table_name: The name of the table to check.
:return: True when the table exists; otherwise, False.
"""
try:
    table = self.dyn_resource.Table(table_name)
    table.load()
    exists = True
except ClientError as err:
    if err.response['Error']['Code'] == 'ResourceNotFoundException':
        exists = False
    else:
        logger.error(
            "Couldn't check for existence of %s. Here's why: %s: %s",
            table_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
else:
    self.table = table
return exists
```

- For API details, see [DescribeTable](#) in *AWS SDK for Python (Boto3) API Reference*.

## List tables

The following code example shows how to list DynamoDB tables.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None

    def list_tables(self):
        """
        Lists the Amazon DynamoDB tables for the current account.

        :return: The list of tables.
        """
        try:
            tables = []
            for table in self.dyn_resource.tables.all():
                print(table.name)
                tables.append(table)
        except ClientError as err:
            logger.error(
                "Couldn't list tables. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return tables
```

- For API details, see [ListTables](#) in *AWS SDK for Python (Boto3) API Reference*.

## Put an item in a table

The following code example shows how to put an item in a DynamoDB table.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:  
    """Encapsulates an Amazon DynamoDB table of movie data."""  
    def __init__(self, dyn_resource):  
        """  
        :param dyn_resource: A Boto3 DynamoDB resource.  
        """  
        self.dyn_resource = dyn_resource  
        self.table = None  
  
    def add_movie(self, title, year, plot, rating):  
        """  
        Adds a movie to the table.  
  
        :param title: The title of the movie.  
        :param year: The release year of the movie.  
        :param plot: The plot summary of the movie.  
        :param rating: The quality rating of the movie.  
        """  
        try:  
            self.table.put_item(  
                Item={  
                    'year': year,  
                    'title': title,  
                    'info': {'plot': plot, 'rating': Decimal(str(rating))}})  
        except ClientError as err:  
            logger.error(  
                "Couldn't add movie %s to table %s. Here's why: %s: %s",  
                title, self.table.name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise
```

- For API details, see [PutItem](#) in *AWS SDK for Python (Boto3) API Reference*.

## Query a table

The following code example shows how to query a DynamoDB table.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Query items by using a key condition expression.

```
class Movies:  
    """Encapsulates an Amazon DynamoDB table of movie data."""
```

```

def __init__(self, dyn_resource):
    """
    :param dyn_resource: A Boto3 DynamoDB resource.
    """
    self.dyn_resource = dyn_resource
    self.table = None

def query_movies(self, year):
    """
    Queries for movies that were released in the specified year.

    :param year: The year to query.
    :return: The list of movies that were released in the specified year.
    """
    try:
        response = self.table.query(KeyConditionExpression=Key('year').eq(year))
    except ClientError as err:
        logger.error(
            "Couldn't query for movies released in %s. Here's why: %s: %s",
            year,
            err.response['Error']['Code'],
            err.response['Error']['Message']
        )
        raise
    else:
        return response['Items']

```

Query items and project them to return a subset of data.

```

class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table

    def query_and_project_movies(self, year, title_bounds):
        """
        Query for movies that were released in a specified year and that have titles
        that start within a range of letters. A projection expression is used
        to return a subset of data for each movie.

        :param year: The release year to query.
        :param title_bounds: The range of starting letters to query.
        :return: The list of movies.
        """
        try:
            response = self.table.query(
                ProjectionExpression="#yr, title, info.genres, info.actors[0]",
                ExpressionAttributeNames={"#yr": "year"},
                KeyConditionExpression=(
                    Key('year').eq(year) &
                    Key('title').between(title_bounds['first'],
                title_bounds['second']))
            )
        except ClientError as err:
            if err.response['Error']['Code'] == "ValidationException":
                logger.warning(
                    "There's a validation error. Here's the message: %s: %s",
                    err.response['Error']['Code'],
                    err.response['Error']['Message']
                )
            else:
                logger.error(
                    "Couldn't query for movies. Here's why: %s: %s",
                    err.response['Error']['Code'],
                    err.response['Error']['Message']
                )
                raise
        else:
            return response['Items']

```

- For API details, see [Query in AWS SDK for Python \(Boto3\) API Reference](#).

## Run a PartiQL statement

The following code example shows how to run a PartiQL statement on a DynamoDB table.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PartiQLWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource

    def run_partiql(self, statement, params):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because the
        resource transforms input and output from plain old Python objects (POPOs) to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statement: The PartiQL statement.
        :param params: The list of PartiQL parameters. These are applied to the
                      statement in the order they are listed.
        :return: The items returned from the statement, if any.
        """
        try:
            output = self.dyn_resource.meta.client.execute_statement(
                Statement=statement, Parameters=params)
        except ClientError as err:
            if err.response['Error']['Code'] == 'ResourceNotFoundException':
                logger.error(
                    "Couldn't execute PartiQL '%s' because the table does not exist.", statement)
            else:
                logger.error(
                    "Couldn't execute PartiQL '%s'. Here's why: %s: %s", statement,
                    err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return output
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Python (Boto3) API Reference*.

## Run batches of PartiQL statements

The following code example shows how to run batches of PartiQL statements on a DynamoDB table.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PartiQLBatchWrapper:  
    """  
    Encapsulates a DynamoDB resource to run PartiQL statements.  
    """  
    def __init__(self, dyn_resource):  
        """  
        :param dyn_resource: A Boto3 DynamoDB resource.  
        """  
        self.dyn_resource = dyn_resource  
  
    def run_partiql(self, statements, param_list):  
        """  
        Runs a PartiQL statement. A Boto3 resource is used even though  
        `execute_statement` is called on the underlying `client` object because the  
        resource transforms input and output from plain old Python objects (POPOs) to  
        the DynamoDB format. If you create the client directly, you must do these  
        transforms yourself.  
  
        :param statements: The batch of PartiQL statements.  
        :param param_list: The batch of PartiQL parameters that are associated with  
                           each statement. This list must be in the same order as the  
                           statements.  
        :return: The responses returned from running the statements, if any.  
        """  
        try:  
            output = self.dyn_resource.meta.client.batch_execute_statement(  
                Statements=[{  
                    'Statement': statement, 'Parameters': params  
                } for statement, params in zip(statements, param_list)])  
        except ClientError as err:  
            if err.response['Error']['Code'] == 'ResourceNotFoundException':  
                logger.error(  
                    "Couldn't execute batch of PartiQL statements because the table "  
                    "does not exist.")  
            else:  
                logger.error(  
                    "Couldn't execute batch of PartiQL statements. Here's why: %s: %s",  
                    err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return output
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scan a table

The following code example shows how to scan a DynamoDB table.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:  
    """Encapsulates an Amazon DynamoDB table of movie data."""  
    def __init__(self, dyn_resource):  
        """  
        :param dyn_resource: A Boto3 DynamoDB resource.  
        """  
        self.dyn_resource = dyn_resource
```

```
    self.table = None

def scan_movies(self, year_range):
    """
    Scans for movies that were released in a range of years.
    Uses a projection expression to return a subset of data for each movie.

    :param year_range: The range of years to retrieve.
    :return: The list of movies released in the specified years.
    """
    movies = []
    scan_kwargs = {
        'FilterExpression': Key('year').between(year_range['first'],
                                              year_range['second']),
        'ProjectionExpression': "#yr, title, info.rating",
        'ExpressionAttributeNames': {"#yr": "year"}}
    try:
        done = False
        start_key = None
        while not done:
            if start_key:
                scan_kwargs['ExclusiveStartKey'] = start_key
            response = self.table.scan(**scan_kwargs)
            movies.extend(response.get('Items', []))
            start_key = response.get('LastEvaluatedKey', None)
            done = start_key is None
    except ClientError as err:
        logger.error(
            "Couldn't scan for movies. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

    return movies
```

- For API details, see [Scan in AWS SDK for Python \(Boto3\) API Reference](#).

## Update an item in a table

The following code example shows how to update an item in a DynamoDB table.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Update an item by using an update expression.

```
class Movies:
    """
    Encapsulates an Amazon DynamoDB table of movie data.
    """
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None

    def update_movie(self, title, year, rating, plot):
        """
        Updates rating and plot data for a movie in the table.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        """
```

```

:param rating: The updated rating to give the movie.
:param plot: The updated plot summary to give the movie.
:return: The fields that were updated, with their new values.
"""
try:
    response = self.table.update_item(
        Key={'year': year, 'title': title},
        UpdateExpression="set info.rating=:r, info.plot=:p",
        ExpressionAttributeValues={
            ':r': Decimal(str(rating)), ':p': plot},
        ReturnValues="UPDATED_NEW")
except ClientError as err:
    logger.error(
        "Couldn't update movie %s in table %s. Here's why: %s: %s",
        title, self.table.name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response['Attributes']

```

Update an item by using an update expression that includes an arithmetic operation.

```

class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table

    def update_rating(self, title, year, rating_change):
        """
        Updates the quality rating of a movie in the table by using an arithmetic
        operation in the update expression. By specifying an arithmetic operation,
        you can adjust a value in a single request, rather than first getting its
        value and then setting its new value.

        :param title: The title of the movie to update.
        :param year: The release year of the movie to update.
        :param rating_change: The amount to add to the current rating for the movie.
        :return: The updated rating.
        """
        try:
            response = self.table.update_item(
                Key={'year': year, 'title': title},
                UpdateExpression="set info.rating = info.rating + :val",
                ExpressionAttributeValues={':val': Decimal(str(rating_change))},
                ReturnValues="UPDATED_NEW")
        except ClientError as err:
            logger.error(
                "Couldn't update movie %s in table %s. Here's why: %s: %s",
                title, self.table.name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['Attributes']

```

Update an item only when it meets certain conditions.

```

class UpdateQueryWrapper:
    def __init__(self, table):
        self.table = table

    def remove_actors(self, title, year, actor_threshold):
        """
        Removes an actor from a movie, but only when the number of actors is greater

```

than a specified threshold. If the movie does not list more than the threshold, no actors are removed.

```
:param title: The title of the movie to update.
:param year: The release year of the movie to update.
:param actor_threshold: The threshold of actors to check.
:return: The movie data after the update.
"""
try:
    response = self.table.update_item(
        Key={'year': year, 'title': title},
        UpdateExpression="remove info.actors[0]",
        ConditionExpression="size(info.actors) > :num",
        ExpressionAttributeValues={':num': actor_threshold},
        ReturnValues="ALL_NEW")
except ClientError as err:
    if err.response['Error']['Code'] == "ConditionalCheckFailedException":
        logger.warning(
            "Didn't update %s because it has fewer than %s actors.", title, actor_threshold + 1)
    else:
        logger.error(
            "Couldn't update movie %s. Here's why: %s: %s", title,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
else:
    return response['Attributes']
```

- For API details, see [UpdateItem](#) in *AWS SDK for Python (Boto3) API Reference*.

## Write a batch of items

The following code example shows how to write a batch of DynamoDB items.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource
        self.table = None

    def write_batch(self, movies):
        """
        Fills an Amazon DynamoDB table with the specified data, using the Boto3
        Table.batch_writer() function to put the items in the table.
        Inside the context manager, Table.batch_writer builds a list of
        requests. On exiting the context manager, Table.batch_writer starts sending
        batches of write requests to Amazon DynamoDB and automatically
        handles chunking, buffering, and retrying.

        :param movies: The data to put in the table. Each item must contain at least
                      the keys required by the schema that was specified when the
                      table was created.
        """
```

```
try:
    with self.table.batch_writer() as writer:
        for movie in movies:
            writer.put_item(Item=movie)
except ClientError as err:
    logger.error(
        "Couldn't load data into table %s. Here's why: %s: %s",
        self.table.name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
```

- For API details, see [BatchWriteItem in AWS SDK for Python \(Boto3\) API Reference](#).

## Scenarios

### Accelerate reads with DAX

The following code example shows how to:

- Create and write data to a table with both the DAX and SDK clients.
- Get, query, and scan the table with both the DAX and SDK clients and compare their performance.

For more information, see [Developing with the DynamoDB Accelerator Client](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a table with either the DAX or Boto3 client.

```
import boto3

def create_dax_table(dyn_resource=None):
    """
    Creates a DynamoDB table.

    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The newly created table.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource('dynamodb')

    table_name = 'TryDaxTable'
    params = {
        'TableName': table_name,
        'KeySchema': [
            {'AttributeName': 'partition_key', 'KeyType': 'HASH'},
            {'AttributeName': 'sort_key', 'KeyType': 'RANGE'}
        ],
        'AttributeDefinitions': [
            {'AttributeName': 'partition_key', 'AttributeType': 'N'},
            {'AttributeName': 'sort_key', 'AttributeType': 'N'}
        ],
        'ProvisionedThroughput': {
            'ReadCapacityUnits': 10,
            'WriteCapacityUnits': 10
        }
    }
```

```
table = dyn_resource.create_table(**params)
print(f"Creating {table_name}...")
table.wait_until_exists()
return table

if __name__ == '__main__':
    dax_table = create_dax_table()
    print(f"Created table.")
```

Write test data to the table.

```
import boto3

def write_data_to_dax_table(key_count, item_size, dyn_resource=None):
    """
    Writes test data to the demonstration table.

    :param key_count: The number of partition and sort keys to use to populate the
                      table. The total number of items is key_count * key_count.
    :param item_size: The size of non-key data for each test item.
    :param dyn_resource: Either a Boto3 or DAX resource.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource('dynamodb')

    table = dyn_resource.Table('TryDaxTable')
    some_data = 'X' * item_size

    for partition_key in range(1, key_count + 1):
        for sort_key in range(1, key_count + 1):
            table.put_item(Item={
                'partition_key': partition_key,
                'sort_key': sort_key,
                'some_data': some_data
            })
    print(f"Put item ({partition_key}, {sort_key}) succeeded.")

if __name__ == '__main__':
    write_key_count = 10
    write_item_size = 1000
    print(f"Writing {write_key_count}*{write_key_count} items to the table. "
          f"Each item is {write_item_size} characters.")
    write_data_to_dax_table(write_key_count, write_item_size)
```

Get items for a number of iterations for both the DAX client and the Boto3 client and report the time spent for each.

```
import argparse
import sys
import time
import amazondax
import boto3

def get_item_test(key_count, iterations, dyn_resource=None):
    """
    Gets items from the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    """
    pass
```

```

and reported.

:param key_count: The number of items to get from the table in each iteration.
:param iterations: The number of iterations to run.
:param dyn_resource: Either a Boto3 or DAX resource.
:return: The start and end times of the test.
"""
if dyn_resource is None:
    dyn_resource = boto3.resource('dynamodb')

table = dyn_resource.Table('TryDaxTable')
start = time.perf_counter()
for _ in range(iterations):
    for partition_key in range(1, key_count + 1):
        for sort_key in range(1, key_count + 1):
            table.get_item(Key={
                'partition_key': partition_key,
                'sort_key': sort_key
            })
            print('.')
            sys.stdout.flush()
print()
end = time.perf_counter()
return start, end

if __name__ == '__main__':
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        'endpoint_url', nargs='?',
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not used.")
    args = parser.parse_args()

    test_key_count = 10
    test_iterations = 50
    if args.endpoint_url:
        print(f"Getting each item from the table {test_iterations} times, "
              f"using the DAX client.")
        # Use a with statement so the DAX client closes the cluster after completion.
        with amazonadax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as dax:
            test_start, test_end = get_item_test(
                test_key_count, test_iterations, dyn_resource=dax)
    else:
        print(f"Getting each item from the table {test_iterations} times, "
              f"using the Boto3 client.")
        test_start, test_end = get_item_test(
            test_key_count, test_iterations)
    print(f"Total time: {test_end - test_start:.4f} sec. Average time: "
          f"{{(test_end - test_start)/ test_iterations}}")

```

Query the table for a number of iterations for both the DAX client and the Boto3 client and report the time spent for each.

```

import argparse
import time
import sys
import amazonadax
import boto3
from boto3.dynamodb.conditions import Key

def query_test(partition_key, sort_keys, iterations, dyn_resource=None):

```

```
"""
Queries the table a specified number of times. The time before the
first iteration and the time after the last iteration are both captured
and reported.

:param partition_key: The partition key value to use in the query. The query
                     returns items that have partition keys equal to this value.
:param sort_keys: The range of sort key values for the query. The query returns
                  items that have sort key values between these two values.
:param iterations: The number of iterations to run.
:param dyn_resource: Either a Boto3 or DAX resource.
:return: The start and end times of the test.
"""

if dyn_resource is None:
    dyn_resource = boto3.resource('dynamodb')

table = dyn_resource.Table('TryDaxTable')
key_condition_expression = \
    Key('partition_key').eq(partition_key) & \
    Key('sort_key').between(*sort_keys)

start = time.perf_counter()
for _ in range(iterations):
    table.query(KeyConditionExpression=key_condition_expression)
    print('.', end='')
    sys.stdout.flush()
print()
end = time.perf_counter()
return start, end

if __name__ == '__main__':
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        'endpoint_url', nargs='?',
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not used.")
    args = parser.parse_args()

    test_partition_key = 5
    test_sort_keys = (2, 9)
    test_iterations = 100
    if args.endpoint_url:
        print(f"Querying the table {test_iterations} times, using the DAX client.")
        # Use a with statement so the DAX client closes the cluster after completion.
        with amazonadax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as dax:
            test_start, test_end = query_test(
                test_partition_key, test_sort_keys, test_iterations, dyn_resource=dax)
    else:
        print(f"Querying the table {test_iterations} times, using the Boto3 client.")
        test_start, test_end = query_test(
            test_partition_key, test_sort_keys, test_iterations)

    print(f"Total time: {test_end - test_start:.4f} sec. Average time: "
          f"{{(test_end - test_start)/test_iterations}}.")
```

Scan the table for a number of iterations for both the DAX client and the Boto3 client and report the time spent for each.

```
import argparse
import time
import sys
import amazonadax
```

```
import boto3

def scan_test(iterations, dyn_resource=None):
    """
    Scans the table a specified number of times. The time before the
    first iteration and the time after the last iteration are both captured
    and reported.

    :param iterations: The number of iterations to run.
    :param dyn_resource: Either a Boto3 or DAX resource.
    :return: The start and end times of the test.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource('dynamodb')

    table = dyn_resource.Table('TryDaxTable')
    start = time.perf_counter()
    for _ in range(iterations):
        table.scan()
        print('.', end='')
        sys.stdout.flush()
    print()
    end = time.perf_counter()
    return start, end

if __name__ == '__main__':
    # pylint: disable=not-context-manager
    parser = argparse.ArgumentParser()
    parser.add_argument(
        'endpoint_url', nargs='?',
        help="When specified, the DAX cluster endpoint. Otherwise, DAX is not used.")
    args = parser.parse_args()

    test_iterations = 100
    if args.endpoint_url:
        print(f"Scanning the table {test_iterations} times, using the DAX client.")
        # Use a with statement so the DAX client closes the cluster after completion.
        with amazondax.AmazonDaxClient.resource(endpoint_url=args.endpoint_url) as dax:
            test_start, test_end = scan_test(test_iterations, dyn_resource=dax)
    else:
        print(f"Scanning the table {test_iterations} times, using the Boto3 client.")
        test_start, test_end = scan_test(test_iterations)
    print(f"Total time: {test_end - test_start:.4f} sec. Average time: "
          f"{{(test_end - test_start)/test_iterations}}.")
```

Delete the table.

```
import boto3

def delete_dax_table(dyn_resource=None):
    """
    Deletes the demonstration table.

    :param dyn_resource: Either a Boto3 or DAX resource.
    """
    if dyn_resource is None:
        dyn_resource = boto3.resource('dynamodb')

    table = dyn_resource.Table('TryDaxTable')
    table.delete()
```

```
    print(f"Deleting {table.name}...")
    table.wait_until_not_exists()

if __name__ == '__main__':
    delete_dax_table()
    print("Table deleted!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateTable](#)
  - [DeleteTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)

## Get started using tables, items, and queries

The following code example shows how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.
- Delete a movie from the table.
- Delete the table.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that encapsulates a DynamoDB table.

```
from decimal import Decimal
from io import BytesIO
import json
import logging
import os
from pprint import pprint
import requests
from zipfile import ZipFile
import boto3
from boto3.dynamodb.conditions import Key
from botocore.exceptions import ClientError
from question import Question

logger = logging.getLogger(__name__)

class Movies:
    """Encapsulates an Amazon DynamoDB table of movie data."""
    def __init__(self, dyn_resource):
        """
```

```
:param dyn_resource: A Boto3 DynamoDB resource.
"""
self.dyn_resource = dyn_resource
self.table = None

def exists(self, table_name):
    """
Determines whether a table exists. As a side effect, stores the table in
a member variable.

:param table_name: The name of the table to check.
:return: True when the table exists; otherwise, False.
"""
try:
    table = self.dyn_resource.Table(table_name)
    table.load()
    exists = True
except ClientError as err:
    if err.response['Error']['Code'] == 'ResourceNotFoundException':
        exists = False
    else:
        logger.error(
            "Couldn't check for existence of %s. Here's why: %s: %s",
            table_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
else:
    self.table = table
return exists

def create_table(self, table_name):
    """
Creates an Amazon DynamoDB table that can be used to store movie data.
The table uses the release year of the movie as the partition key and the
title as the sort key.

:param table_name: The name of the table to create.
:return: The newly created table.
"""
try:
    self.table = self.dyn_resource.create_table(
        TableName=table_name,
        KeySchema=[
            {'AttributeName': 'year', 'KeyType': 'HASH'}, # Partition key
            {'AttributeName': 'title', 'KeyType': 'RANGE'} # Sort key
        ],
        AttributeDefinitions=[
            {'AttributeName': 'year', 'AttributeType': 'N'},
            {'AttributeName': 'title', 'AttributeType': 'S'}
        ],
        ProvisionedThroughput={'ReadCapacityUnits': 10, 'WriteCapacityUnits':
10})
    self.table.wait_until_exists()
except ClientError as err:
    logger.error(
        "Couldn't create table %s. Here's why: %s: %s", table_name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return self.table

def list_tables(self):
    """
Lists the Amazon DynamoDB tables for the current account.

:return: The list of tables.
```

```
"""
try:
    tables = []
    for table in self.dyn_resource.tables.all():
        print(table.name)
        tables.append(table)
except ClientError as err:
    logger.error(
        "Couldn't list tables. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return tables

def write_batch(self, movies):
    """
    Fills an Amazon DynamoDB table with the specified data, using the Boto3
    Table.batch_writer() function to put the items in the table.
    Inside the context manager, Table.batch_writer builds a list of
    requests. On exiting the context manager, Table.batch_writer starts sending
    batches of write requests to Amazon DynamoDB and automatically
    handles chunking, buffering, and retrying.

    :param movies: The data to put in the table. Each item must contain at least
                   the keys required by the schema that was specified when the
                   table was created.
    """
    try:
        with self.table.batch_writer() as writer:
            for movie in movies:
                writer.put_item(Item=movie)
    except ClientError as err:
        logger.error(
            "Couldn't load data into table %s. Here's why: %s: %s",
            self.table.name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

    def add_movie(self, title, year, plot, rating):
        """
        Adds a movie to the table.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        :param plot: The plot summary of the movie.
        :param rating: The quality rating of the movie.
        """
        try:
            self.table.put_item(
                Item={
                    'year': year,
                    'title': title,
                    'info': {'plot': plot, 'rating': Decimal(str(rating))}})
        except ClientError as err:
            logger.error(
                "Couldn't add movie %s to table %s. Here's why: %s: %s",
                title, self.table.name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise

    def get_movie(self, title, year):
        """
        Gets movie data from the table for a specific movie.

        :param title: The title of the movie.
        :param year: The release year of the movie.
        """
```

```

:return: The data about the requested movie.
"""
try:
    response = self.table.get_item(Key={'year': year, 'title': title})
except ClientError as err:
    logger.error(
        "Couldn't get movie %s from table %s. Here's why: %s: %s",
        title, self.table.name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response['Item']

def update_movie(self, title, year, rating, plot):
    """
    Updates rating and plot data for a movie in the table.

    :param title: The title of the movie to update.
    :param year: The release year of the movie to update.
    :param rating: The updated rating to give the movie.
    :param plot: The updated plot summary to give the movie.
    :return: The fields that were updated, with their new values.
    """
    try:
        response = self.table.update_item(
            Key={'year': year, 'title': title},
            UpdateExpression="set info.rating=:r, info.plot=:p",
            ExpressionAttributeValues={
                ':r': Decimal(str(rating)), ':p': plot},
            ReturnValues="UPDATED_NEW")
    except ClientError as err:
        logger.error(
            "Couldn't update movie %s in table %s. Here's why: %s: %s",
            title, self.table.name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response['Attributes']

def query_movies(self, year):
    """
    Queries for movies that were released in the specified year.

    :param year: The year to query.
    :return: The list of movies that were released in the specified year.
    """
    try:
        response = self.table.query(KeyConditionExpression=Key('year').eq(year))
    except ClientError as err:
        logger.error(
            "Couldn't query for movies released in %s. Here's why: %s: %s",
            year, err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response['Items']

def scan_movies(self, year_range):
    """
    Scans for movies that were released in a range of years.
    Uses a projection expression to return a subset of data for each movie.

    :param year_range: The range of years to retrieve.
    :return: The list of movies released in the specified years.
    """
    movies = []
    scan_kwargs = {

```

```

        'FilterExpression': Key('year').between(year_range['first'],
year_range['second']),
        'ProjectionExpression': "#yr, title, info.rating",
        'ExpressionAttributeNames': {"#yr": "year"}}

    try:
        done = False
        start_key = None
        while not done:
            if start_key:
                scan_kwargs['ExclusiveStartKey'] = start_key
            response = self.table.scan(**scan_kwargs)
            movies.extend(response.get('Items', []))
            start_key = response.get('LastEvaluatedKey', None)
            done = start_key is None
    except ClientError as err:
        logger.error(
            "Couldn't scan for movies. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

    return movies

def delete_movie(self, title, year):
    """
    Deletes a movie from the table.

    :param title: The title of the movie to delete.
    :param year: The release year of the movie to delete.
    """
    try:
        self.table.delete_item(Key={'year': year, 'title': title})
    except ClientError as err:
        logger.error(
            "Couldn't delete movie %s. Here's why: %s: %s",
            title, err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def delete_table(self):
    """
    Deletes the table.

    """
    try:
        self.table.delete()
        self.table = None
    except ClientError as err:
        logger.error(
            "Couldn't delete table. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

```

Create a helper function to download and extract the sample JSON file.

```

def get_sample_movie_data(movie_file_name):
    """
    Gets sample movie data, either from a local file or by first downloading it from
    the Amazon DynamoDB developer guide.

    :param movie_file_name: The local file name where the movie data is stored in JSON
    format.
    :return: The movie data as a dict.
    """
    if not os.path.isfile(movie_file_name):
        print(f"Downloading {movie_file_name}...")
        movie_content = requests.get(

```

```
'https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/moviedata.zip')
movie_zip = ZipFile(BytesIO(movie_content.content))
movie_zip.extractall()

try:
    with open(movie_file_name) as movie_file:
        movie_data = json.load(movie_file, parse_float=Decimal)
except FileNotFoundError:
    print(f"File {movie_file_name} not found. You must first download the file to "
          "run this demo. See the README for instructions.")
    raise
else:
    # The sample file lists over 4000 movies, return only the first 250.
    return movie_data[:250]
```

Run an interactive scenario to create the table and perform actions on it.

```
def run_scenario(table_name, movie_file_name, dyn_resource):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the Amazon DynamoDB getting started demo.")
    print('*'*88)

    movies = Movies(dyn_resource)
    movies_exists = movies.exists(table_name)
    if not movies_exists:
        print(f"\nCreating table {table_name}...")
        movies.create_table(table_name)
        print(f"\nCreated table {movies.table.name}.")

    my_movie = Question.ask_questions([
        Question('title', "Enter the title of a movie you want to add to the table: "),
        Question('year', "What year was it released? ", Question.is_int),
        Question(
            'rating', "On a scale of 1 - 10, how do you rate it? ",
            Question.is_float, Question.in_range(1, 10)),
        Question('plot', "Summarize the plot for me: ")
    ])
    movies.add_movie(**my_movie)
    print(f"\nAdded '{my_movie['title']}' to '{movies.table.name}'")
    print('*'*88)

    movie_update = Question.ask_questions([
        Question(
            'rating',
            f"\nLet's update your movie.\nYou rated it {my_movie['rating']}, what new "
            f"rating would you give it? ", Question.is_float, Question.in_range(1,
10)),
        Question(
            'plot',
            f"You summarized the plot as '{my_movie['plot']}'.\nWhat would you say now?
")]
    )
    my_movie.update(movie_update)
    updated = movies.update_movie(**my_movie)
    print(f"\nUpdated '{my_movie['title']}' with new attributes:")
    pprint(updated)
    print('*'*88)

    if not movies_exists:
        movie_data = get_sample_movie_data(movie_file_name)
        print(f"\nReading data from '{movie_file_name}' into your table.")
        movies.write_batch(movie_data)
```

```
    print(f"\nWrote {len(movie_data)} movies into {movies.table.name}.")
print('*'*88)

    title = "The Lord of the Rings: The Fellowship of the Ring"
    if Question.ask_question(
        f"Let's move on...do you want to get info about '{title}'? (y/n) ",
        Question.is_yesno):
        movie = movies.get_movie(title, 2001)
        print("\nHere's what I found:")
        pprint(movie)
    print('*'*88)

    ask_for_year = True
    while ask_for_year:
        release_year = Question.ask_question(
            f"\nLet's get a list of movies released in a given year. Enter a year
between "
            f"1972 and 2018: ", Question.is_int, Question.in_range(1972, 2018))
        releases = movies.query_movies(release_year)
        if releases:
            print(f"There were {len(releases)} movies released in {release_year}:")
            for release in releases:
                print(f"\t{release['title']}")
            ask_for_year = False
        else:
            print(f"I don't know about any movies released in {release_year}!")
            ask_for_year = Question.ask_question("Try another year? (y/n) ",
Question.is_yesno)
    print('*'*88)

    years = Question.ask_questions([
        Question(
            'first',
            f"\nNow let's scan for movies released in a range of years. Enter a year:
",
            Question.is_int, Question.in_range(1972, 2018)),
        Question(
            'second', "Now enter another year: ",
            Question.is_int, Question.in_range(1972, 2018)))
    releases = movies.scan_movies(years)
    if releases:
        count = Question.ask_question(
            f"\nFound {len(releases)} movies. How many do you want to see? ",
            Question.is_int, Question.in_range(1, len(releases)))
        print(f"\nHere are your {count} movies:\n")
        pprint(releases[:count])
    else:
        print(f"I don't know about any movies released between {years['first']} "
              f"and {years['second']}.")
    print('*'*88)

    if Question.ask_question(
        f"\nLet's remove your movie from the table. Do you want to remove "
        f"'{my_movie['title']}'? (y/n)", Question.is_yesno):
        movies.delete_movie(my_movie['title'], my_movie['year'])
        print(f"\nRemoved '{my_movie['title']}' from the table.")
    print('*'*88)

    if Question.ask_question(f"\nDelete the table? (y/n) ", Question.is_yesno):
        movies.delete_table()
        print(f"Deleted {table_name}.")
    else:
        print("Don't forget to delete the table when you're done or you might incur "
              "charges on your account.")

print("\nThanks for watching!")
```

```
print('*'*88)

if __name__ == '__main__':
    try:
        run_scenario(
            'doc-example-table-movies', 'moviedata.json', boto3.resource('dynamodb'))
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")
```

This scenario uses the following helper class to ask questions at a command prompt.

```
class Question:
    """
    A helper class to ask questions at a command prompt and validate and convert
    the answers.
    """
    def __init__(self, key, question, *validators):
        """
        :param key: The key that is used for storing the answer in a dict, when
                   multiple questions are asked in a set.
        :param question: The question to ask.
        :param validators: The answer is passed through the list of validators until
                           one fails or they all pass. Validators may also convert the
                           answer to another form, such as from a str to an int.
        """
        self.key = key
        self.question = question
        self.validators = Question.non_empty, *validators

    @staticmethod
    def ask_questions(questions):
        """
        Asks a set of questions and stores the answers in a dict.

        :param questions: The list of questions to ask.
        :return: A dict of answers.
        """
        answers = {}
        for question in questions:
            answers[question.key] = Question.ask_question(
                question.question, *question.validators)
        return answers

    @staticmethod
    def ask_question(question, *validators):
        """
        Asks a single question and validates it against a list of validators.
        When an answer fails validation, the complaint is printed and the question
        is asked again.

        :param question: The question to ask.
        :param validators: The list of validators that the answer must pass.
        :return: The answer, converted to its final form by the validators.
        """
        answer = None
        while answer is None:
            answer = input(question)
            for validator in validators:
                answer, complaint = validator(answer)
                if answer is None:
                    print(complaint)
                    break
        return answer
```

```
@staticmethod
def non_empty(answer):
    """
        Validates that the answer is not empty.
        :return: The non-empty answer, or None.
    """
    return answer if answer != '' else None, "I need an answer. Please?"

@staticmethod
def is_yesno(answer):
    """
        Validates a yes/no answer.
        :return: True when the answer is 'y'; otherwise, False.
    """
    return answer.lower() == 'y', ""

@staticmethod
def is_int(answer):
    """
        Validates that the answer can be converted to an int.
        :return: The int answer; otherwise, None.
    """
    try:
        int_answer = int(answer)
    except ValueError:
        int_answer = None
    return int_answer, f"{answer} must be a valid integer."

@staticmethod
def is_letter(answer):
    """
        Validates that the answer is a letter.
        :return: The letter answer, converted to uppercase; otherwise, None.
    """
    return answer.upper() if answer.isalpha() else None, f"{answer} must be a single letter."

@staticmethod
def is_float(answer):
    """
        Validate that the answer can be converted to a float.
        :return: The float answer; otherwise, None.
    """
    try:
        float_answer = float(answer)
    except ValueError:
        float_answer = None
    return float_answer, f"{answer} must be a valid float."

@staticmethod
def in_range(lower, upper):
    """
        Validate that the answer is within a range. The answer must be of a type that
        can
        be compared to the lower and upper bounds.
        :return: The answer, if it is within the range; otherwise, None.
    """
    def _validate(answer):
        return (
            answer if lower <= answer <= upper else None,
            f"{answer} must be between {lower} and {upper}.")
    return _validate
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## Query a table by using batches of PartiQL statements

The following code example shows how to:

- Get a batch of items by running multiple SELECT statements.
- Add a batch of items by running multiple INSERT statements.
- Update a batch of items by running multiple UPDATE statements.
- Delete a batch of items by running multiple DELETE statements.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that can run batches of PartiQL statements.

```
from datetime import datetime
from decimal import Decimal
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

from scaffold import Scaffold

logger = logging.getLogger(__name__)

class PartiQLBatchWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource

    def run_partiql(self, statements, param_list):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        'execute_statement' is called on the underlying 'client' object because the
        resource transforms input and output from plain old Python objects (POPOs) to
        the DynamoDB format. If you create the client directly, you must do these
        """


```

```

transforms yourself.

:param statements: The batch of PartiQL statements.
:param param_list: The batch of PartiQL parameters that are associated with
                   each statement. This list must be in the same order as the
                   statements.
:return: The responses returned from running the statements, if any.
"""
try:
    output = self.dyn_resource.meta.client.batch_execute_statement(
        Statements=[{
            'Statement': statement, 'Parameters': params
            } for statement, params in zip(statements, param_list)])
except ClientError as err:
    if err.response['Error']['Code'] == 'ResourceNotFoundException':
        logger.error(
            "Couldn't execute batch of PartiQL statements because the table "
            "does not exist.")
    else:
        logger.error(
            "Couldn't execute batch of PartiQL statements. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return output

```

Run a scenario that creates a table and runs PartiQL queries in batches.

```

def run_scenario(scaffold, wrapper, table_name):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the Amazon DynamoDB PartiQL batch statement demo.")
    print('*'*88)

    print(f"Creating table '{table_name}' for the demo...")
    scaffold.create_table(table_name)
    print('*'*88)

    movie_data = [
        {
            'title': f"House PartiQL",
            'year': datetime.now().year - 5,
            'info': {
                'plot': "Wacky high jinks result from querying a mysterious database.",
                'rating': Decimal('8.5')}},
        {
            'title': f"House PartiQL 2",
            'year': datetime.now().year - 3,
            'info': {
                'plot': "Moderate high jinks result from querying another mysterious
database.",
                'rating': Decimal('6.5')}},
        {
            'title': f"House PartiQL 3",
            'year': datetime.now().year - 1,
            'info': {
                'plot': "Tepid high jinks result from querying yet another mysterious
database.",
                'rating': Decimal('2.5')}}

    print(f"Inserting a batch of movies into table '{table_name}.")
    statements = [
        f"INSERT INTO \\"{table_name}\\" "
        f"VALUE {{'title': ?, 'year': ?, 'info': ?}}"] * len(movie_data)
    params = [list(movie.values()) for movie in movie_data]
    wrapper.run_partiql(statements, params)

```

```

print("Success!")
print('*'*88)

print(f"Getting data for a batch of movies.")
statements = [
    f"SELECT * FROM \'{table_name}\' WHERE title=? AND year=?"] * len(movie_data)
params = [[movie['title'], movie['year']] for movie in movie_data]
output = wrapper.run_partiql(statements, params)
for item in output['Responses']:
    print(f"\n{item['Item']['title']}, {item['Item']['year']}")
    pprint(item['Item'])
print('*'*88)

ratings = [Decimal('7.7'), Decimal('5.5'), Decimal('1.3')]
print(f"Updating a batch of movies with new ratings.")
statements = [
    f"UPDATE \'{table_name}\' SET info.rating=? "
    f"WHERE title=? AND year=?"] * len(movie_data)
params = [
    [rating, movie['title'], movie['year']] for rating, movie in zip(ratings,
movie_data)]
wrapper.run_partiql(statements, params)
print("Success!")
print('*'*88)

print(f"Getting projected data from the table to verify our update.")
output = wrapper.dyn_resource.meta.client.execute_statement(
    Statement=f"SELECT title, info.rating FROM '{table_name}'")
pprint(output['Items'])
print('*'*88)

print(f"Deleting a batch of movies from the table.")
statements = [
    f"DELETE FROM \'{table_name}\' WHERE title=? AND year=?"] * len(movie_data)
params = [[movie['title'], movie['year']] for movie in movie_data]
wrapper.run_partiql(statements, params)
print("Success!")
print('*'*88)

print(f"Deleting table '{table_name}'...")
scaffold.delete_table()
print('*'*88)

print("\nThanks for watching!")
print('*'*88)

if __name__ == '__main__':
    try:
        dyn_res = boto3.resource('dynamodb')
        scaffold = Scaffold(dyn_res)
        movies = PartiQLBatchWrapper(dyn_res)
        run_scenario(scaffold, movies, 'doc-example-table-partiql-movies')
    except Exception as e:
        print(f"Something went wrong with the demo! Here's what: {e}")

```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Python (Boto3) API Reference*.

## Query a table using PartiQL

The following code example shows how to:

- Get an item by running a SELECT statement.

- Add an item by running an INSERT statement.
- Update an item by running an UPDATE statement.
- Delete an item by running a DELETE statement.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that can run PartiQL statements.

```
from datetime import datetime
from decimal import Decimal
import logging
from pprint import pprint

import boto3
from botocore.exceptions import ClientError

from scaffold import Scaffold

logger = logging.getLogger(__name__)

class PartiQLWrapper:
    """
    Encapsulates a DynamoDB resource to run PartiQL statements.
    """
    def __init__(self, dyn_resource):
        """
        :param dyn_resource: A Boto3 DynamoDB resource.
        """
        self.dyn_resource = dyn_resource

    def run_partiql(self, statement, params):
        """
        Runs a PartiQL statement. A Boto3 resource is used even though
        `execute_statement` is called on the underlying `client` object because the
        resource transforms input and output from plain old Python objects (POPOs) to
        the DynamoDB format. If you create the client directly, you must do these
        transforms yourself.

        :param statement: The PartiQL statement.
        :param params: The list of PartiQL parameters. These are applied to the
                      statement in the order they are listed.
        :return: The items returned from the statement, if any.
        """
        try:
            output = self.dyn_resource.meta.client.execute_statement(
                Statement=statement, Parameters=params)
        except ClientError as err:
            if err.response['Error']['Code'] == 'ResourceNotFoundException':
                logger.error(
                    "Couldn't execute PartiQL '%s' because the table does not exist.",
                    statement)
            else:
                logger.error(
                    "Couldn't execute PartiQL '%s'. Here's why: %s: %s",
                    statement,
                    err.response['Error']['Code'],
                    err.response['Error']['Message'])
            raise
        else:
            return output
```

Run a scenario that creates a table and runs PartiQL queries.

```
def run_scenario(scaffold, wrapper, table_name):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the Amazon DynamoDB PartiQL single statement demo.")
    print('*'*88)

    print(f"Creating table '{table_name}' for the demo...")
    scaffold.create_table(table_name)
    print('*'*88)

    title = "24 Hour PartiQL People"
    year = datetime.now().year
    plot = "A group of data developers discover a new query language they can't stop
using."
    rating = Decimal('9.9')

    print(f"Inserting movie '{title}' released in {year}.")
    wrapper.run_partiql(
        f"INSERT INTO \"'{table_name}'\" VALUE {{'title': ?, 'year': ?, 'info': ?}},",
        [title, year, {'plot': plot, 'rating': rating}])
    print("Success!")
    print('*'*88)

    print(f"Getting data for movie '{title}' released in {year}.")
    output = wrapper.run_partiql(
        f"SELECT * FROM \"'{table_name}'\" WHERE title=? AND year=?", [title, year])
    for item in output['Items']:
        print(f"\n{item['title']}, {item['year']}")  

        pprint(output['Items'])
    print('*'*88)

    rating = Decimal('2.4')
    print(f"Updating movie '{title}' with a rating of {float(rating)}.")
    wrapper.run_partiql(
        f"UPDATE \"'{table_name}'\" SET info.rating=? WHERE title=? AND year=?",
        [rating, title, year])
    print("Success!")
    print('*'*88)

    print(f"Getting data again to verify our update.")
    output = wrapper.run_partiql(
        f"SELECT * FROM \"'{table_name}'\" WHERE title=? AND year=?", [title, year])
    for item in output['Items']:
        print(f"\n{item['title']}, {item['year']}")  

        pprint(output['Items'])
    print('*'*88)

    print(f"Deleting movie '{title}' released in {year}.")
    wrapper.run_partiql(
        f"DELETE FROM \"'{table_name}'\" WHERE title=? AND year=?",
        [title, year])
    print("Success!")
    print('*'*88)

    print(f"Deleting table '{table_name}'...")
    scaffold.delete_table()
    print('*'*88)

    print("\nThanks for watching!")
    print('*'*88)

if __name__ == '__main__':
```

```
try:
    dyn_res = boto3.resource('dynamodb')
    scaffold = Scaffold(dyn_res)
    movies = PartiQLWrapper(dyn_res)
    run_scenario(scaffold, movies, 'doc-example-table-partiql-movies')
except Exception as e:
    print(f"Something went wrong with the demo! Here's what: {e}")
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Python (Boto3) API Reference*.

## Amazon EC2 examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Elastic Compute Cloud.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Get started

#### [Hello Amazon EC2](#)

The following code examples show how to get started using Amazon Elastic Compute Cloud (Amazon EC2).

Java

#### [SDK for Java 2.x](#)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public static void describeEC2SecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
            DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId)
                .build();

        DescribeSecurityGroupsResponse response =
            ec2.describeSecurityGroups(request);
        for(SecurityGroup group : response.securityGroups()) {
            System.out.printf(
                "Found Security Group with id %s, " +
                "vpc id %s " +
                "and description %s",
                group.groupId(),
                group.vpcId(),
                group.description());
        }
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for Java 2.x API Reference*.

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import boto3

def hello_ec2(ec2_resource):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Elastic Compute Cloud
    (Amazon EC2) resource and list the security groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ec2_resource: A Boto3 EC2 ServiceResource object. This object is a high-
    level
                           resource that wraps the low-level EC2 service API.
    """
    print("Hello, Amazon EC2! Let's list up to 10 of your security groups:")
    for sg in ec2_resource.security_groups.limit(10):
        print(f"\t{sg.id}: {sg.group_name}")

if __name__ == '__main__':
    hello_ec2(boto3.resource('ec2'))
```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for Python (Boto3) API Reference*.

## Topics

- [Actions \(p. 3895\)](#)
- [Scenarios \(p. 3911\)](#)

## Actions

### Allocate an Elastic IP address

The following code example shows how to allocate an Elastic IP address for Amazon EC2.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
actions."""
    def __init__(self, ec2_resource, elastic_ip=None):
        """
```

```
:param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                    is used to create additional high-level objects
                    that wrap low-level Amazon EC2 service actions.
:param elastic_ip: A Boto3 VpcAddress object. This is a high-level object that
                    wraps Elastic IP actions.
"""
self.ec2_resource = ec2_resource
self.elastic_ip = elastic_ip

@classmethod
def from_resource(cls):
    ec2_resource = boto3.resource('ec2')
    return cls(ec2_resource)

def allocate(self):
    """
    Allocates an Elastic IP address that can be associated with an Amazon EC2
    instance. By using an Elastic IP address, you can keep the public IP address
    constant even when you restart the associated instance.

    :return: The newly created Elastic IP object. By default, the address is not
            associated with any instance.
    """
    try:
        response = self.ec2_resource.meta.client.allocate_address(Domain='vpc')
        self.elastic_ip = self.ec2_resource.VpcAddress(response['AllocationId'])
    except ClientError as err:
        logger.error(
            "Couldn't allocate Elastic IP. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return self.elastic_ip
```

- For API details, see [AllocateAddress in AWS SDK for Python \(Boto3\) API Reference](#).

## Associate an Elastic IP address with an instance

The following code example shows how to associate an Elastic IP address with an Amazon EC2 instance.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
    actions."""
    def __init__(self, ec2_resource, elastic_ip=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param elastic_ip: A Boto3 VpcAddress object. This is a high-level object that
                            wraps Elastic IP actions.
        """
        self.ec2_resource = ec2_resource
        self.elastic_ip = elastic_ip

    @classmethod
```

```

def from_resource(cls):
    ec2_resource = boto3.resource('ec2')
    return cls(ec2_resource)

def associate(self, instance):
    """
    Associates an Elastic IP address with an instance. When this association is
    created, the Elastic IP's public IP address is immediately used as the public
    IP address of the associated instance.

    :param instance: A Boto3 Instance object. This is a high-level object that
    wraps
                    Amazon EC2 instance actions.
    :return: A response that contains the ID of the association.
    """
    if self.elastic_ip is None:
        logger.info("No Elastic IP to associate.")
        return

    try:
        response = self.elastic_ip.associate(InstanceId=instance.id)
    except ClientError as err:
        logger.error(
            "Couldn't associate Elastic IP %s with instance %s. Here's why: %s:",
            self.elastic_ip.allocation_id, instance.id,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    return response

```

- For API details, see [AssociateAddress](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a security group

The following code example shows how to create an Amazon EC2 security group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
    actions."""
    def __init__(self, ec2_resource, security_group=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param security_group: A Boto3 SecurityGroup object. This is a high-level
        object
                            that wraps security group actions.
        """
        self.ec2_resource = ec2_resource
        self.security_group = security_group

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

```

```
def create(self, group_name, group_description):
    """
    Creates a security group in the default virtual private cloud (VPC) of the
    current account.

    :param group_name: The name of the security group to create.
    :param group_description: The description of the security group to create.
    :return: A Boto3 SecurityGroup object that represents the newly created
    security group.
    """
    try:
        self.security_group = self.ec2_resource.create_security_group(
            GroupName=group_name, Description=group_description)
    except ClientError as err:
        logger.error(
            "Couldn't create security group %s. Here's why: %s: %s",
            group_name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return self.security_group
```

- For API details, see [CreateSecurityGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a security key pair

The following code example shows how to create a security key pair for Amazon EC2.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyPairWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions."""
    def __init__(self, ec2_resource, key_file_dir, key_pair=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param key_file_dir: The folder where the private key information is stored.
                            This should be a secure folder.
        :param key_pair: A Boto3 KeyPair object. This is a high-level object that
                        wraps key pair actions.
        """
        self.ec2_resource = ec2_resource
        self.key_pair = key_pair
        self.key_file_path = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource, tempfile.TemporaryDirectory())

    def create(self, key_name):
        """
        Creates a key pair that can be used to securely connect to an EC2 instance.
        The returned key pair contains private key information that cannot be retrieved
        again. The private key data is stored as a .pem file.
        """

```

```
:param key_name: The name of the key pair to create.  
:return: A Boto3 KeyPair object that represents the newly created key pair.  
"""  
try:  
    self.key_pair = self.ec2_resource.create_key_pair(KeyName=key_name)  
    self.key_file_path = os.path.join(self.key_file_dir.name,  
f'{self.key_pair.name}.pem')  
    with open(self.key_file_path, 'w') as key_file:  
        key_file.write(self.key_pair.key_material)  
except ClientError as err:  
    logger.error(  
        "Couldn't create key %s. Here's why: %s: %s", key_name,  
        err.response['Error']['Code'], err.response['Error']['Message'])  
    raise  
else:  
    return self.key_pair
```

- For API details, see [CreateKeyPair](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create and run an instance

The following code example shows how to create and run an Amazon EC2 instance.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:  
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""  
    def __init__(self, ec2_resource, instance=None):  
        """  
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource  
                           is used to create additional high-level objects  
                           that wrap low-level Amazon EC2 service actions.  
        :param instance: A Boto3 Instance object. This is a high-level object that  
                        wraps instance actions.  
        """  
        self.ec2_resource = ec2_resource  
        self.instance = instance  
  
    @classmethod  
    def from_resource(cls):  
        ec2_resource = boto3.resource('ec2')  
        return cls(ec2_resource)  
  
    def create(  
        self, image, instance_type, key_pair, security_groups=None):  
        """  
        Creates a new EC2 instance. The instance starts immediately after  
        it is created.  
  
        The instance is created in the default VPC of the current account.  
        :param image: A Boto3 Image object that represents an Amazon Machine Image  
        (AMI)  
                           that defines attributes of the instance that is created. The AMI  
                           defines things like the kind of operating system and the type of  
                           storage used by the instance.  
        """
```

```

:param instance_type: The type of instance to create, such as 't2.micro'.
                     The instance type defines things like the number of CPUs
and
                     the amount of memory.
:param key_pair: A Boto3 KeyPair or KeyPairInfo object that represents the key
                     pair that is used to secure connections to the instance.
:param security_groups: A list of Boto3 SecurityGroup objects that represents
the
                     security groups that are used to grant access to the
                     instance. When no security groups are specified, the
                     default security group of the VPC is used.
:return: A Boto3 Instance object that represents the newly created instance.
"""
try:
    instance_params = {
        'ImageId': image.id, 'InstanceType': instance_type, 'KeyName':
key_pair.name
    }
    if security_groups is not None:
        instance_params['SecurityGroupIds'] = [sg.id for sg in security_groups]
    self.instance = self.ec2_resource.create_instances(**instance_params,
MinCount=1, MaxCount=1)[0]
    self.instance.wait_until_running()
except ClientError as err:
    logging.error(
        "Couldn't create instance with image %s, instance type %s, and key %s.
"
        "Here's why: %s: %s", image.id, instance_type, key_pair.name,
err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return self.instance

```

- For API details, see [RunInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a security group

The following code example shows how to delete an Amazon EC2 security group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
actions."""
    def __init__(self, ec2_resource, security_group=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                           is used to create additional high-level objects
                           that wrap low-level Amazon EC2 service actions.
        :param security_group: A Boto3 SecurityGroup object. This is a high-level
object
                           that wraps security group actions.
        """
        self.ec2_resource = ec2_resource
        self.security_group = security_group

    @classmethod

```

```
def from_resource(cls):
    ec2_resource = boto3.resource('ec2')
    return cls(ec2_resource)

def delete(self):
    """
    Deletes the security group.
    """
    if self.security_group is None:
        logger.info("No security group to delete.")
        return

    group_id = self.security_group.id
    try:
        self.security_group.delete()
    except ClientError as err:
        logger.error(
            "Couldn't delete security group %s. Here's why: %s: %s",
            group_id,
            err.response['Error']['Code'],
            err.response['Error']['Message']
        )
        raise
```

- For API details, see [DeleteSecurityGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a security key pair

The following code example shows how to delete an Amazon EC2 security key pair.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyPairWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions."""
    def __init__(self, ec2_resource, key_file_dir, key_pair=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param key_file_dir: The folder where the private key information is stored.
                            This should be a secure folder.
        :param key_pair: A Boto3 KeyPair object. This is a high-level object that
                        wraps key pair actions.
        """
        self.ec2_resource = ec2_resource
        self.key_pair = key_pair
        self.key_file_path = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource, tempfile.TemporaryDirectory())

    def delete(self):
        """
        Deletes a key pair.
        """
        if self.key_pair is None:
            logger.info("No key pair to delete.")
```

```
        return

    key_name = self.key_pair.name
    try:
        self.key_pair.delete()
        self.key_pair = None
    except ClientError as err:
        logger.error(
            "Couldn't delete key %s. Here's why: %s : %s",
            key_name,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
```

- For API details, see [DeleteKeyPair](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe instances

The following code example shows how to describe Amazon EC2 instances.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""
    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object that
                        wraps instance actions.
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def display(self, indent=1):
        """
        Displays information about an instance.

        :param indent: The visual indent to apply to the output.
        """
        if self.instance is None:
            logger.info("No instance to display.")
            return

        try:
            self.instance.load()
            ind = '\t'*indent
            print(f"{ind}ID: {self.instance.id}")
            print(f"{ind}Image ID: {self.instance.image_id}")
            print(f"{ind}Instance type: {self.instance.instance_type}")
            print(f"{ind}Key name: {self.instance.key_name}")
            print(f"{ind}VPC ID: {self.instance.vpc_id}")
            print(f"{ind}Public IP: {self.instance.public_ip_address}")
        
```

```
        print(f"{{ind}}State: {self.instance.state['Name']}")
    except ClientError as err:
        logger.error(
            "Couldn't display your instance. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
```

- For API details, see [DescribeInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Disassociate an Elastic IP address from an instance

The following code example shows how to disassociate an Elastic IP address from an Amazon EC2 instance.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
    actions."""
    def __init__(self, ec2_resource, elastic_ip=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param elastic_ip: A Boto3 VpcAddress object. This is a high-level object that
                           wraps Elastic IP actions.
        """
        self.ec2_resource = ec2_resource
        self.elastic_ip = elastic_ip

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def disassociate(self):
        """
        Removes an association between an Elastic IP address and an instance. When the
        association is removed, the instance is assigned a new public IP address.
        """
        if self.elastic_ip is None:
            logger.info("No Elastic IP to disassociate.")
            return

        try:
            self.elastic_ip.association.delete()
        except ClientError as err:
            logger.error(
                "Couldn't disassociate Elastic IP %s from its instance. Here's why: %s: %s",
                self.elastic_ip.allocation_id,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

- For API details, see [DisassociateAddress](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get data about Amazon Machine Images

The following code example shows how to get data about Amazon Machine Images (AMIs).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""
    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object that
                        wraps instance actions.
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def get_images(self, image_ids):
        """
        Gets information about Amazon Machine Images (AMIs) from a list of AMI IDs.

        :param image_ids: The list of AMIs to look up.
        :return: A list of Boto3 Image objects that represent the requested AMIs.
        """
        try:
            images = list(self.ec2_resource.images.filter(ImageIds=image_ids))
        except ClientError as err:
            logger.error(
                "Couldn't get images. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return images
```

- For API details, see [DescribeImages in AWS SDK for Python \(Boto3\) API Reference](#).

## Get data about a security group

The following code example shows how to get data about an Amazon EC2 security group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
    actions."""
```

```

def __init__(self, ec2_resource, security_group=None):
    """
    :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                        is used to create additional high-level objects
                        that wrap low-level Amazon EC2 service actions.
    :param security_group: A Boto3 SecurityGroup object. This is a high-level
                          object
                                         that wraps security group actions.
    """
    self.ec2_resource = ec2_resource
    self.security_group = security_group

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def describe(self):
        """
        Displays information about the security group.
        """
        if self.security_group is None:
            logger.info("No security group to describe.")
            return

        try:
            print(f"Security group: {self.security_group.group_name}")
            print(f"\tID: {self.security_group.id}")
            print(f"\tVPC: {self.security_group.vpc_id}")
            if self.security_group.ip_permissions:
                print(f"\tInbound permissions:")
                pp(self.security_group.ip_permissions)
        except ClientError as err:
            logger.error(
                "Couldn't get data for security group %s. Here's why: %s: %s",
                self.security_group.id,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise

```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get data about instance types

The following code example shows how to get data about Amazon EC2 instance types.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""
    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object that
                         wraps instance actions.
        """

```

```
self.ec2_resource = ec2_resource
self.instance = instance

@classmethod
def from_resource(cls):
    ec2_resource = boto3.resource('ec2')
    return cls(ec2_resource)

def get_instance_types(self, architecture):
    """
    Gets instance types that support the specified architecture and are designated
    as either 'micro' or 'small'. When an instance is created, the instance type
    you specify must support the architecture of the AMI you use.

    :param architecture: The kind of architecture the instance types must support,
                         such as 'x86_64'.
    :return: A list of instance types that support the specified architecture
             and are either 'micro' or 'small'.
    """
    try:
        inst_types = []
        it Paginator =
self.ec2_resource.meta.client.getPaginator('describe_instance_types')
        for page in it Paginator.paginate(
            Filters=[{
                'Name': 'processor-info.supported-architecture', 'Values':
[architecture],
                {'Name': 'instance-type', 'Values': ['*.micro', '*.small']}])
            inst_types += page['InstanceTypes']
    except ClientError as err:
        logger.error(
            "Couldn't get instance types. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return inst_types
```

- For API details, see [DescribeInstanceTypes](#) in *AWS SDK for Python (Boto3) API Reference*.

## List security key pairs

The following code example shows how to list Amazon EC2 security key pairs.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyPairWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions."""
    def __init__(self, ec2_resource, key_file_dir, key_pair=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param key_file_dir: The folder where the private key information is stored.
                            This should be a secure folder.
        :param key_pair: A Boto3 KeyPair object. This is a high-level object that
                        wraps key pair actions.
        """
```

```
self.ec2_resource = ec2_resource
self.key_pair = key_pair
self.key_file_path = None
self.key_file_dir = key_file_dir

@classmethod
def from_resource(cls):
    ec2_resource = boto3.resource('ec2')
    return cls(ec2_resource, tempfile.TemporaryDirectory())

def list(self, limit):
    """
    Displays a list of key pairs for the current account.

    :param limit: The maximum number of key pairs to list.
    """
    try:
        for kp in self.ec2_resource.key_pairs.limit(limit):
            print(f"Found {kp.key_type} key {kp.name} with fingerprint:")
            print(f"\t{kp.key_fingerprint}")
    except ClientError as err:
        logger.error(
            "Couldn't list key pairs. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
```

- For API details, see [DescribeKeyPairs](#) in *AWS SDK for Python (Boto3) API Reference*.

## Release an Elastic IP address

The following code example shows how to release an Elastic IP address.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
    actions."""
    def __init__(self, ec2_resource, elastic_ip=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param elastic_ip: A Boto3 VpcAddress object. This is a high-level object that
                           wraps Elastic IP actions.
        """
        self.ec2_resource = ec2_resource
        self.elastic_ip = elastic_ip

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def release(self):
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
        it can no longer be used.
```

```
"""
if self.elastic_ip is None:
    logger.info("No Elastic IP to release.")
    return

try:
    self.elastic_ip.release()
except ClientError as err:
    logger.error(
        "Couldn't release Elastic IP address %s. Here's why: %s: %s",
        self.elastic_ip.allocation_id,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
```

- For API details, see [ReleaseAddress in AWS SDK for Python \(Boto3\) API Reference](#).

## Set inbound rules for a security group

The following code example shows how to set inbound rules for an Amazon EC2 security group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
    actions."""
    def __init__(self, ec2_resource, security_group=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param security_group: A Boto3 SecurityGroup object. This is a high-level
                            object
                            that wraps security group actions.
        """
        self.ec2_resource = ec2_resource
        self.security_group = security_group

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def authorize_ingress(self, ssh_ingress_ip):
        """
        Adds a rule to the security group to allow access to SSH.

        :param ssh_ingress_ip: The IP address that is granted inbound access to connect
                              to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of the
                response indicates whether the request succeeded or failed.
        """
        if self.security_group is None:
            logger.info("No security group to update.")
            return

        try:
            ip_permissions = [{
```

```
# SSH ingress open to only the specified IP address.
'IpProtocol': 'tcp', 'FromPort': 22, 'ToPort': 22,
'IpRanges': [{('CidrIp': f'{ssh_ingress_ip}/32')}]]

response =
self.security_group.authorize_ingress(IpPermissions=ip_permissions)
except ClientError as err:
    logger.error(
        "Couldn't authorize inbound rules for %s. Here's why: %s: %s",
        self.security_group.id,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response
```

- For API details, see [AuthorizeSecurityGroupIngress](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start an instance

The following code example shows how to start an Amazon EC2 instance.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""
    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object that
                        wraps instance actions.
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def start(self):
        """
        Starts an instance and waits for it to be in a running state.

        :return: The response to the start request.
        """
        if self.instance is None:
            logger.info("No instance to start.")
            return

        try:
            response = self.instance.start()
            self.instance.wait_until_running()
        except ClientError as err:
            logger.error(
                "Couldn't start instance %s. Here's why: %s: %s", self.instance.id,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

```
    else:  
        return response
```

- For API details, see [StartInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Stop an instance

The following code example shows how to stop an Amazon EC2 instance.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:  
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""  
    def __init__(self, ec2_resource, instance=None):  
        """  
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource  
            is used to create additional high-level objects  
            that wrap low-level Amazon EC2 service actions.  
        :param instance: A Boto3 Instance object. This is a high-level object that  
            wraps instance actions.  
        """  
        self.ec2_resource = ec2_resource  
        self.instance = instance  
  
    @classmethod  
    def from_resource(cls):  
        ec2_resource = boto3.resource('ec2')  
        return cls(ec2_resource)  
  
    def stop(self):  
        """  
        Stops an instance and waits for it to be in a stopped state.  
        :return: The response to the stop request.  
        """  
        if self.instance is None:  
            logger.info("No instance to stop.")  
            return  
  
        try:  
            response = self.instance.stop()  
            self.instance.wait_until_stopped()  
        except ClientError as err:  
            logger.error(  
                "Couldn't stop instance %s. Here's why: %s: %s", self.instance.id,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return response
```

- For API details, see [StopInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Terminate an instance

The following code example shows how to terminate an Amazon EC2 instance.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""
    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object that
                        wraps instance actions.
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def terminate(self):
        """
        Terminates an instance and waits for it to be in a terminated state.
        """
        if self.instance is None:
            logger.info("No instance to terminate.")
            return

        instance_id = self.instance.id
        try:
            self.instance.terminate()
            self.instance.wait_until_terminated()
            self.instance = None
        except ClientError as err:
            logging.error(
                "Couldn't terminate instance %s. Here's why: %s: %s",
                instance_id,
                err.response['Error']['Code'],
                err.response['Error']['Message']
            )
            raise
```

- For API details, see [TerminateInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Get started with instances

The following code example shows how to:

- Create a key pair that is used to secure SSH communication between your computer and an EC2 instance.
- Create a security group that acts as a virtual firewall for your EC2 instances to control incoming and outgoing traffic.
- Find an Amazon Machine Image (AMI) and a compatible instance type.
- Create an instance that is created from the instance type and AMI you select, and is configured to use the security group and key pair created in this example.

- Stop and restart the instance.
- Create an Elastic IP address and associate it as a consistent IP address for your instance.
- Connect to your instance with SSH, using both its public IP address and your Elastic IP address.
- Clean up all of the resources created by this example.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
class Ec2InstanceScenario:  
    """Runs an interactive scenario that shows how to get started using EC2  
    instances."""  
    def __init__(self, inst_wrapper, key_wrapper, sg_wrapper, eip_wrapper, ssm_client):  
        """  
        :param inst_wrapper: An object that wraps instance actions.  
        :param key_wrapper: An object that wraps key pair actions.  
        :param sg_wrapper: An object that wraps security group actions.  
        :param eip_wrapper: An object that wraps Elastic IP actions.  
        :param ssm_client: A Boto3 AWS Systems Manager client.  
        """  
        self.inst_wrapper = inst_wrapper  
        self.key_wrapper = key_wrapper  
        self.sg_wrapper = sg_wrapper  
        self.eip_wrapper = eip_wrapper  
        self.ssm_client = ssm_client  
  
    @demo_func  
    def create_and_list_key_pairs(self):  
        """  
        1. Creates an RSA key pair and saves its private key data as a .pem file in  
        secure  
            temporary storage. The private key data is deleted after the example  
        completes.  
        2. Lists the first five key pairs for the current account.  
        """  
        print("Let's create an RSA key pair that you can be use to securely connect to  
"  
             "your EC2 instance.")  
        key_name = q.ask("Enter a unique name for your key: ", q.non_empty)  
        self.key_wrapper.create(key_name)  
        print(f"Created a key pair {self.key_wrapper.key_pair.key_name} and saved the "  
             f"private key to {self.key_wrapper.key_file_path}.\\n")  
        if q.ask("Do you want to list some of your key pairs? (y/n) ", q.is_yesno):  
            self.key_wrapper.list(5)  
  
    @demo_func  
    def create_security_group(self):  
        """  
        1. Creates a security group for the default VPC.  
        2. Adds an inbound rule to allow SSH. The SSH rule allows only  
            inbound traffic from the current computer's public IPv4 address.  
        3. Displays information about the security group.  
  
        This function uses 'http://checkip.amazonaws.com' to get the current public IP  
        address of the computer that is running the example. This method works in most  
        cases. However, depending on how your computer connects to the internet, you  
        might have to manually add your public IP address to the security group by  
        using  
            the AWS Management Console.  
        """
```

```

"""
print("Let's create a security group to manage access to your instance.")
sg_name = q.ask("Enter a unique name for your security group: ", q.non_empty)
security_group = self.sg_wrapper.create(
    sg_name, "Security group for example: get started with instances.")
print(f"Created security group {security_group.group_name} in your default "
      f"VPC {security_group.vpc_id}.\n")

ip_response = urllib.request.urlopen('http://checkip.amazonaws.com')
current_ip_address = ip_response.read().decode('utf-8').strip()
print("Let's add a rule to allow SSH only from your current IP address.")
print(f"Your public IP address is {current_ip_address}.")
q.ask("Press Enter to add this rule to your security group.")
response = self.sg_wrapper.authorize_ingress(current_ip_address)
if response['Return']:
    print("Security group rules updated.")
else:
    print("Couldn't update security group rules.")
self.sg_wrapper.describe()

@demo_func
def create_instance(self):
    """
    1. Gets a list of Amazon Linux 2 AMIs from AWS Systems Manager. Specifying the
       '/aws/service/ami-amazon-linux-latest' path returns only the latest AMIs.
    2. Gets and displays information about the available AMIs and lets you select
       one.
    3. Gets a list of instance types that are compatible with the selected AMI and
       lets you select one.
    4. Creates an instance with the previously created key pair and security group,
       and the selected AMI and instance type.
    5. Waits for the instance to be running and then displays its information.
    """
    amiPaginator = self.ssm_client.get Paginator('get_parameters_by_path')
    amiOptions = []
    for page in amiPaginator.paginate(Path='/aws/service/ami-amazon-linux-
latest'):
        amiOptions += page['Parameters']
    amzn2Images = self.inst_wrapper.get_images(
        [opt['Value'] for opt in amiOptions if 'amzn2' in opt['Name']])
    print("Let's create an instance from an Amazon Linux 2 AMI. Here are some
options:")
    imageChoice = q.choose(
        "Which one do you want to use? ", [opt.description for opt in
amzn2Images])
    print("Great choice!\n")

    print(f"Here are some instance types that support the "
          f"{amzn2Images[imageChoice].architecture} architecture of the image:")
    instTypes =
    self.inst_wrapper.get_instance_types(amzn2Images[imageChoice].architecture)
    instTypeChoice = q.choose(
        "Which one do you want to use? ", [it['InstanceType'] for it in
instTypes])
    print("Another great choice.\n")

    print("Creating your instance and waiting for it to start...")
    self.inst_wrapper.create(
        amzn2Images[imageChoice],
        instTypes[instTypeChoice]['InstanceType'],
        self.key_wrapper.key_pair,
        [self.sg_wrapper.security_group])
    print(f"Your instance is ready:\n")
    self.inst_wrapper.display()

    print("You can use SSH to connect to your instance.")

```

```

        print("If the connection attempt times out, you might have to manually update "
              "the SSH ingress rule for your IP address in the AWS Management
Console.")
        self._display_ssh_info()

    def _display_ssh_info(self):
        """
        Displays an SSH connection string that can be used to connect to a running
        instance.
        """
        print("To connect, open another command prompt and run the following command:")
        if self.eip_wrapper.elastic_ip is None:
            print(f"\tssh -i {self.key_wrapper.key_file_path} "
                  f"ec2-user@{self.inst_wrapper.instance.public_ip_address}")
        else:
            print(f"\tssh -i {self.key_wrapper.key_file_path} "
                  f"ec2-user@{self.eip_wrapper.elastic_ip.public_ip}")
        q.ask("Press Enter when you're ready to continue the demo.")

    @demo_func
    def associate_elastic_ip(self):
        """
        1. Allocates an Elastic IP address and associates it with the instance.
        2. Displays an SSH connection string that uses the Elastic IP address.
        """
        print("You can allocate an Elastic IP address and associate it with your
instance\n"
              "to keep a consistent IP address even when your instance restarts.")
        elastic_ip = self.eip_wrapper.allocate()
        print(f"Allocated static Elastic IP address: {elastic_ip.public_ip}.")
        self.eip_wrapper.associate(self.inst_wrapper.instance)
        print(f"Associated your Elastic IP with your instance.")
        print("You can now use SSH to connect to your instance by using the Elastic
IP.")
        self._display_ssh_info()

    @demo_func
    def stop_and_start_instance(self):
        """
        1. Stops the instance and waits for it to stop.
        2. Starts the instance and waits for it to start.
        3. Displays information about the instance.
        4. Displays an SSH connection string. When an Elastic IP address is associated
           with the instance, the IP address stays consistent when the instance stops
           and starts.
        """
        print("Let's stop and start your instance to see what changes.")
        print("Stopping your instance and waiting until it's stopped...")
        self.inst_wrapper.stop()
        print("Your instance is stopped. Restarting...")
        self.inst_wrapper.start()
        print("Your instance is running.")
        self.inst_wrapper.display()
        if self.eip_wrapper.elastic_ip is None:
            print("Every time your instance is restarted, its public IP address
changes.")
        else:
            print("Because you have associated an Elastic IP with your instance, you
can \n"
                  "connect by using a consistent IP address after the instance
restarts.")
        self._display_ssh_info()

    @demo_func
    def cleanup(self):
        """

```

```

1. Disassociate and delete the previously created Elastic IP.
2. Terminate the previously created instance.
3. Delete the previously created security group.
4. Delete the previously created key pair.
"""
print("Let's clean everything up. This example created these resources:")
print(f"\tElastic IP: {self.eip_wrapper.elastic_ip.allocation_id}")
print(f"\tInstance: {self.inst_wrapper.instance.id}")
print(f"\tSecurity group: {self.sg_wrapper.security_group.id}")
print(f"\tKey pair: {self.key_wrapper.key_pair.name}")
if q.ask("Ready to delete these resources? (y/n) ", q.is_yesno):
    self.eip_wrapper.disassociate()
    print("Disassociated the Elastic IP from the instance.")
    self.eip_wrapper.release()
    print("Released the Elastic IP.")
    print("Terminating the instance and waiting for it to terminate...")
    self.inst_wrapper.terminate()
    print("Instance terminated.")
    self.sg_wrapper.delete()
    print("Deleted security group.")
    self.key_wrapper.delete()
    print("Deleted key pair.")

def run_scenario(self):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started
with instances demo.")
    print('*'*88)

    self.create_and_list_key_pairs()
    self.create_security_group()
    self.create_instance()
    self.stop_and_start_instance()
    self.associate_elastic_ip()
    self.stop_and_start_instance()
    self.cleanup()

    print("\nThanks for watching!")
    print('*'*88)

if __name__ == '__main__':
    try:
        scenario = Ec2InstanceScenario(
            InstanceWrapper.from_resource(), KeyPairWrapper.from_resource(),
            SecurityGroupWrapper.from_resource(), ElasticIpWrapper.from_resource(),
            boto3.client('ssm'))
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")

```

Define a class that wraps key pair actions.

```

class KeyPairWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions."""
    def __init__(self, ec2_resource, key_file_dir, key_pair=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param key_file_dir: The folder where the private key information is stored.
                            This should be a secure folder.

```

```
:param key_pair: A Boto3 KeyPair object. This is a high-level object that
                  wraps key pair actions.
"""
self.ec2_resource = ec2_resource
self.key_pair = key_pair
self.key_file_path = None
self.key_file_dir = key_file_dir

@classmethod
def from_resource(cls):
    ec2_resource = boto3.resource('ec2')
    return cls(ec2_resource, tempfile.TemporaryDirectory())

def create(self, key_name):
    """
Creates a key pair that can be used to securely connect to an EC2 instance.
The returned key pair contains private key information that cannot be retrieved
again. The private key data is stored as a .pem file.

:param key_name: The name of the key pair to create.
:return: A Boto3 KeyPair object that represents the newly created key pair.
"""
try:
    self.key_pair = self.ec2_resource.create_key_pair(KeyName=key_name)
    self.key_file_path = os.path.join(self.key_file_dir.name,
f'{self.key_pair.name}.pem')
    with open(self.key_file_path, 'w') as key_file:
        key_file.write(self.key_pair.key_material)
except ClientError as err:
    logger.error(
        "Couldn't create key %s. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return self.key_pair

def list(self, limit):
    """
Displays a list of key pairs for the current account.

:param limit: The maximum number of key pairs to list.
"""
try:
    for kp in self.ec2_resource.key_pairs.limit(limit):
        print(f"Found {kp.key_type} key {kp.name} with fingerprint:")
        print(f"\t{kp.key_fingerprint}")
except ClientError as err:
    logger.error(
        "Couldn't list key pairs. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise

def delete(self):
    """
Deletes a key pair.
"""
if self.key_pair is None:
    logger.info("No key pair to delete.")
    return

key_name = self.key_pair.name
try:
    self.key_pair.delete()
    self.key_pair = None
except ClientError as err:
    logger.error(
```

```
"Couldn't delete key %s. Here's why: %s : %s", key_name,
err.response['Error']['Code'], err.response['Error']['Message'])
raise
```

Define a class that wraps security group actions.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
    actions."""
    def __init__(self, ec2_resource, security_group=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param security_group: A Boto3 SecurityGroup object. This is a high-level
                            object
                            that wraps security group actions.
        """
        self.ec2_resource = ec2_resource
        self.security_group = security_group

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def create(self, group_name, group_description):
        """
        Creates a security group in the default virtual private cloud (VPC) of the
        current account.

        :param group_name: The name of the security group to create.
        :param group_description: The description of the security group to create.
        :return: A Boto3 SecurityGroup object that represents the newly created
                security group.
        """
        try:
            self.security_group = self.ec2_resource.create_security_group(
                GroupName=group_name, Description=group_description)
        except ClientError as err:
            logger.error(
                "Couldn't create security group %s. Here's why: %s : %s",
                group_name, err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return self.security_group

    def authorize_ingress(self, ssh_ingress_ip):
        """
        Adds a rule to the security group to allow access to SSH.

        :param ssh_ingress_ip: The IP address that is granted inbound access to connect
                              to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of the
                response indicates whether the request succeeded or failed.
        """
        if self.security_group is None:
            logger.info("No security group to update.")
            return

        try:
            ip_permissions = [
                # SSH ingress open to only the specified IP address.
                {'IpProtocol': 'tcp', 'FromPort': 22, 'ToPort': 22,
```

```

        'IpRanges': [ {'CidrIp': f'{ssh_ingress_ip}/32'}]]]
    response =
self.security_group.authorize_ingress(IpPermissions=ip_permissions)
except ClientError as err:
    logger.error(
        "Couldn't authorize inbound rules for %s. Here's why: %s: %s",
        self.security_group.id,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response

def describe(self):
    """
    Displays information about the security group.
    """
    if self.security_group is None:
        logger.info("No security group to describe.")
        return

    try:
        print(f"Security group: {self.security_group.group_name}")
        print(f"\tID: {self.security_group.id}")
        print(f"\tVPC: {self.security_group.vpc_id}")
        if self.security_group.ip_permissions:
            print(f"\t\tInbound permissions:")
            pp(self.security_group.ip_permissions)
    except ClientError as err:
        logger.error(
            "Couldn't get data for security group %s. Here's why: %s: %s",
            self.security_group.id,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def delete(self):
    """
    Deletes the security group.
    """
    if self.security_group is None:
        logger.info("No security group to delete.")
        return

    group_id = self.security_group.id
    try:
        self.security_group.delete()
    except ClientError as err:
        logger.error(
            "Couldn't delete security group %s. Here's why: %s: %s",
            group_id,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

```

Define a class that wraps instance actions.

```

class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions."""
    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object that
                        wraps instance actions.
        """
        self.ec2_resource = ec2_resource

```

```

        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource('ec2')
        return cls(ec2_resource)

    def create(
            self, image, instance_type, key_pair, security_groups=None):
        """
        Creates a new EC2 instance. The instance starts immediately after
        it is created.

        The instance is created in the default VPC of the current account.

        :param image: A Boto3 Image object that represents an Amazon Machine Image
        (AMI)
                that defines attributes of the instance that is created. The AMI
                defines things like the kind of operating system and the type of
                storage used by the instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
                The instance type defines things like the number of CPUs
        and
                the amount of memory.
        :param key_pair: A Boto3 KeyPair or KeyPairInfo object that represents the key
                pair that is used to secure connections to the instance.
        :param security_groups: A list of Boto3 SecurityGroup objects that represents
        the
                security groups that are used to grant access to the
                instance. When no security groups are specified, the
                default security group of the VPC is used.
        :return: A Boto3 Instance object that represents the newly created instance.
        """
        try:
            instance_params = {
                'ImageId': image.id, 'InstanceType': instance_type, 'KeyName':
key_pair.name
            }
            if security_groups is not None:
                instance_params['SecurityGroupIds'] = [sg.id for sg in security_groups]
            self.instance = self.ec2_resource.create_instances(**instance_params,
MinCount=1, MaxCount=1)[0]
            self.instance.wait_until_running()
        except ClientError as err:
            logging.error(
                "Couldn't create instance with image %s, instance type %s, and key %s.
"
                "Here's why: %s: %s", image.id, instance_type, key_pair.name,
err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return self.instance
    def display(self, indent=1):
        """
        Displays information about an instance.

        :param indent: The visual indent to apply to the output.
        """
        if self.instance is None:
            logger.info("No instance to display.")
            return
        try:
            self.instance.load()
            ind = '\t'*indent

```

```
print(f"{ind}ID: {self.instance.id}")
print(f"{ind}Image ID: {self.instance.image_id}")
print(f"{ind}Instance type: {self.instance.instance_type}")
print(f"{ind}Key name: {self.instance.key_name}")
print(f"{ind}VPC ID: {self.instance.vpc_id}")
print(f"{ind}Public IP: {self.instance.public_ip_address}")
print(f"{ind}State: {self.instance.state['Name']}")")
except ClientError as err:
    logger.error(
        "Couldn't display your instance. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise

def terminate(self):
    """
    Terminates an instance and waits for it to be in a terminated state.
    """
    if self.instance is None:
        logger.info("No instance to terminate.")
        return

    instance_id = self.instance.id
    try:
        self.instance.terminate()
        self.instance.wait_until_terminated()
        self.instance = None
    except ClientError as err:
        logging.error(
            "Couldn't terminate instance %s. Here's why: %s: %s",
            instance_id, err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def start(self):
    """
    Starts an instance and waits for it to be in a running state.

    :return: The response to the start request.
    """
    if self.instance is None:
        logger.info("No instance to start.")
        return

    try:
        response = self.instance.start()
        self.instance.wait_until_running()
    except ClientError as err:
        logger.error(
            "Couldn't start instance %s. Here's why: %s: %s",
            self.instance.id, err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response

def stop(self):
    """
    Stops an instance and waits for it to be in a stopped state.

    :return: The response to the stop request.
    """
    if self.instance is None:
        logger.info("No instance to stop.")
        return

    try:
        response = self.instance.stop()
        self.instance.wait_until_stopped()
```

```

except ClientError as err:
    logger.error(
        "Couldn't stop instance %s. Here's why: %s: %s", self.instance.id,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response

def get_images(self, image_ids):
    """
    Gets information about Amazon Machine Images (AMIs) from a list of AMI IDs.

    :param image_ids: The list of AMIs to look up.
    :return: A list of Boto3 Image objects that represent the requested AMIs.
    """
    try:
        images = list(self.ec2_resource.images.filter(ImageIds=image_ids))
    except ClientError as err:
        logger.error(
            "Couldn't get images. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return images

def get_instance_types(self, architecture):
    """
    Gets instance types that support the specified architecture and are designated
    as either 'micro' or 'small'. When an instance is created, the instance type
    you specify must support the architecture of the AMI you use.

    :param architecture: The kind of architecture the instance types must support,
                         such as 'x86_64'.
    :return: A list of instance types that support the specified architecture
            and are either 'micro' or 'small'.
    """
    try:
        inst_types = []
        it Paginator =
self.ec2_resource.meta.client.getPaginator('describe_instance_types')
        for page in itPaginator.paginate(
            Filters=[{
                'Name': 'processor-info.supported-architecture', 'Values':
[architecture],
                {'Name': 'instance-type', 'Values': ['*.micro', '*.small']}])
            inst_types += page['InstanceTypes']
    except ClientError as err:
        logger.error(
            "Couldn't get instance types. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return inst_types

```

Define a class that wraps Elastic IP actions.

```

class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
actions."""
    def __init__(self, ec2_resource, elastic_ip=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.

```

```
:param elastic_ip: A Boto3 VpcAddress object. This is a high-level object that
                      wraps Elastic IP actions.
"""
self.ec2_resource = ec2_resource
self.elastic_ip = elastic_ip

@classmethod
def from_resource(cls):
    ec2_resource = boto3.resource('ec2')
    return cls(ec2_resource)

def allocate(self):
    """
    Allocates an Elastic IP address that can be associated with an Amazon EC2
    instance. By using an Elastic IP address, you can keep the public IP address
    constant even when you restart the associated instance.

    :return: The newly created Elastic IP object. By default, the address is not
             associated with any instance.
    """
try:
    response = self.ec2_resource.meta.client.allocate_address(Domain='vpc')
    self.elastic_ip = self.ec2_resource.VpcAddress(response['AllocationId'])
except ClientError as err:
    logger.error(
        "Couldn't allocate Elastic IP. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return self.elastic_ip

def associate(self, instance):
    """
    Associates an Elastic IP address with an instance. When this association is
    created, the Elastic IP's public IP address is immediately used as the public
    IP address of the associated instance.

    :param instance: A Boto3 Instance object. This is a high-level object that
                     wraps
                     Amazon EC2 instance actions.
    :return: A response that contains the ID of the association.
    """
if self.elastic_ip is None:
    logger.info("No Elastic IP to associate.")
    return

try:
    response = self.elastic_ip.associate(InstanceId=instance.id)
except ClientError as err:
    logger.error(
        "Couldn't associate Elastic IP %s with instance %s. Here's why: %s:",
        self.elastic_ip.allocation_id, instance.id,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
return response

def disassociate(self):
    """
    Removes an association between an Elastic IP address and an instance. When the
    association is removed, the instance is assigned a new public IP address.
    """
if self.elastic_ip is None:
    logger.info("No Elastic IP to disassociate.")
    return
```

```
try:
    self.elastic_ip.association.delete()
except ClientError as err:
    logger.error(
        "Couldn't disassociate Elastic IP %s from its instance. Here's why: %s",
        self.elastic_ip.allocation_id,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise

def release(self):
    """
    Releases an Elastic IP address. After the Elastic IP address is released,
    it can no longer be used.
    """
    if self.elastic_ip is None:
        logger.info("No Elastic IP to release.")
        return

    try:
        self.elastic_ip.release()
    except ClientError as err:
        logger.error(
            "Couldn't release Elastic IP address %s. Here's why: %s: %s",
            self.elastic_ip.allocation_id,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [AllocateAddress](#)
  - [AssociateAddress](#)
  - [AuthorizeSecurityGroupIngress](#)
  - [CreateKeyPair](#)
  - [CreateSecurityGroup](#)
  - [DeleteKeyPair](#)
  - [DeleteSecurityGroup](#)
  - [DescribeImages](#)
  - [DescribeInstanceTypes](#)
  - [DescribeInstances](#)
  - [DescribeKeyPairs](#)
  - [DescribeSecurityGroups](#)
  - [DisassociateAddress](#)
  - [ReleaseAddress](#)
  - [RunInstances](#)
  - [StartInstances](#)
  - [StopInstances](#)
  - [TerminateInstances](#)

## Amazon EC2 Auto Scaling examples using SDK for Python (Boto3)

---

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon EC2 Auto Scaling.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3924\)](#)
- [Scenarios \(p. 3930\)](#)

## Actions

### Create a group

The following code example shows how to create an Auto Scaling group.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:  
    """Encapsulates Amazon EC2 Auto Scaling actions."""  
    def __init__(self, autoscaling_client):  
        """  
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.  
        """  
        self.autoscaling_client = autoscaling_client  
  
    def create_group(  
        self, group_name, group_zones, launch_template_name, min_size, max_size):  
        """  
        Creates an Auto Scaling group.  
  
        :param group_name: The name to give to the group.  
        :param group_zones: The Availability Zones in which instances can be created.  
        :param launch_template_name: The name of an existing Amazon EC2 launch  
        template.  
                    The launch template specifies the configuration of  
                    instances that are created by auto scaling  
        activities.  
        :param min_size: The minimum number of active instances in the group.  
        :param max_size: The maximum number of active instances in the group.  
        """  
        try:  
            self.autoscaling_client.create_auto_scaling_group(  
                AutoScalingGroupName=group_name,  
                AvailabilityZones=group_zones,  
                LaunchTemplate={  
                    'LaunchTemplateName': launch_template_name, 'Version': '$Default'},  
                MinSize=min_size, MaxSize=max_size  
            )  
        except ClientError as err:  
            logger.error(  
                "Couldn't create group %s. Here's why: %s: %s", group_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise
```

- For API details, see [CreateAutoScalingGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a group

The following code example shows how to delete an Auto Scaling group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:  
    """Encapsulates Amazon EC2 Auto Scaling actions."""  
    def __init__(self, autoscaling_client):  
        """  
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.  
        """  
        self.autoscaling_client = autoscaling_client  
  
    def delete_group(self, group_name):  
        """  
        Deletes an Auto Scaling group. All instances must be stopped before the  
        group can be deleted.  
  
        :param group_name: The name of the group to delete.  
        """  
        try:  
            self.autoscaling_client.delete_auto_scaling_group(  
                AutoScalingGroupName=group_name)  
        except ClientError as err:  
            logger.error(  
                "Couldn't delete group %s. Here's why: %s: %s", group_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise
```

- For API details, see [DeleteAutoScalingGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Disable metrics collection for a group

The following code example shows how to disable CloudWatch metrics collection for an Auto Scaling group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:  
    """Encapsulates Amazon EC2 Auto Scaling actions."""  
    def __init__(self, autoscaling_client):  
        """  
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.  
        """  
        self.autoscaling_client = autoscaling_client  
  
    def disable_metrics(self, group_name):  
        """  
        Stops CloudWatch metric collection for the Auto Scaling group.  
  
        :param group_name: The name of the group.  
        """
```

```
"""
try:
    self.autoscaling_client.disable_metrics_collection(
        AutoScalingGroupName=group_name)
except ClientError as err:
    logger.error(
        "Couldn't disable metrics %s. Here's why: %s: %s",
        group_name,
        err.response['Error']['Code'],
        err.response['Error']['Message'])
    raise
```

- For API details, see [DisableMetricsCollection](#) in *AWS SDK for Python (Boto3) API Reference*.

## Enable metrics collection for a group

The following code example shows how to enable CloudWatch metrics collection for an Auto Scaling group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def enable_metrics(self, group_name, metrics):
        """
        Enables CloudWatch metric collection for Amazon EC2 Auto Scaling activities.

        :param group_name: The name of the group to enable.
        :param metrics: A list of metrics to collect.
        """
        try:
            self.autoscaling_client.enable_metrics_collection(
                AutoScalingGroupName=group_name, Metrics=metrics,
                Granularity='1Minute')
        except ClientError as err:
            logger.error(
                "Couldn't enable metrics on %s. Here's why: %s: %s",
                group_name,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
```

- For API details, see [EnableMetricsCollection](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get information about groups

The following code example shows how to get information about Auto Scaling groups.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def describe_group(self, group_name):
        """
        Gets information about an Auto Scaling group.

        :param group_name: The name of the group to look up.
        :return: Information about the group, if found.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name])
        except ClientError as err:
            logger.error(
                "Couldn't describe group %s. Here's why: %s: %s",
                group_name,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
        else:
            groups = response.get('AutoScalingGroups', [])
            return groups[0] if len(groups) > 0 else None
```

- For API details, see [DescribeAutoScalingGroups](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get information about instances

The following code example shows how to get information about Auto Scaling instances.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def describe_instances(self, instance_ids):
        """
        Gets information about instances.

        :param instance_ids: A list of instance IDs to look up.
        :return: Information about instances, or an empty list if none are found.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_instances(
                InstanceIds=instance_ids)
        except ClientError as err:
            logger.error(
                "Couldn't describe instances %s. Here's why: %s: %s",
                instance_ids,
```

```
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response['AutoScalingInstances']
```

- For API details, see [DescribeAutoScalingInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get information about scaling activities

The following code example shows how to get information about Auto Scaling activities.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def describe_scaling_activities(self, group_name):
        """
        Gets information about scaling activities for the group. Scaling activities
        are things like instances stopping or starting in response to user requests
        or capacity changes.

        :param group_name: The name of the group to look up.
        :return: The list of scaling activities for the group, ordered with the most
                recent activity first.
        """
        try:
            response = self.autoscaling_client.describe_scaling_activities(
                AutoScalingGroupName=group_name)
        except ClientError as err:
            logger.error(
                "Couldn't describe scaling activities %s. Here's why: %s: %s",
                group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['Activities']
```

- For API details, see [DescribeScalingActivities](#) in *AWS SDK for Python (Boto3) API Reference*.

## Set the desired capacity of a group

The following code example shows how to set the desired capacity of an Auto Scaling group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def set_desired_capacity(self, group_name, capacity):
        """
        Sets the desired capacity of the group. Amazon EC2 Auto Scaling tries to keep
        the
        number of running instances equal to the desired capacity.

        :param group_name: The name of the group to update.
        :param capacity: The desired number of running instances.
        """
        try:
            self.autoscaling_client.set_desired_capacity(
                AutoScalingGroupName=group_name, DesiredCapacity=capacity,
                HonorCooldown=False)
        except ClientError as err:
            logger.error(
                "Couldn't set desired capacity %s. Here's why: %s: %s",
                group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

- For API details, see [SetDesiredCapacity](#) in *AWS SDK for Python (Boto3) API Reference*.

## Terminate an instance in a group

The following code example shows how to terminate an instance in an Auto Scaling group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def terminate_instance(self, instance_id, decrease_capacity):
        """
        Stops an instance.

        :param instance_id: The ID of the instance to stop.
        :param decrease_capacity: Specifies whether to decrease the desired capacity
                                  of the group. When passing True for this parameter,
                                  you can stop an instance without having a replacement
                                  instance start when the desired capacity threshold is
                                  crossed.
        :return: The scaling activity that occurs in response to this action.
        """
        try:
```

```
        response =
    self.autoscaling_client.terminate_instance_in_auto_scaling_group(
        InstanceId=instance_id,
        ShouldDecrementDesiredCapacity=decrease_capacity)
    except ClientError as err:
        logger.error(
            "Couldn't terminate instance %s. Here's why: %s: %s",
            instance_id,
            err.response['Error']['Code'],
            err.response['Error']['Message'])
        raise
    else:
        return response['Activity']
```

- For API details, see [TerminateInstanceInAutoScalingGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update a group

The following code example shows how to update the configuration for an Auto Scaling group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client

    def update_group(self, group_name, **kwargs):
        """
        Updates an Auto Scaling group.

        :param group_name: The name of the group to update.
        :param kwargs: Keyword arguments to pass through to the service.
        """
        try:
            self.autoscaling_client.update_auto_scaling_group(
                AutoScalingGroupName=group_name, **kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't update group %s. Here's why: %s: %s",
                group_name,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
```

- For API details, see [UpdateAutoScalingGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Manage groups and instances

The following code example shows how to:

- Create an Amazon EC2 Auto Scaling group and configure it with a launch template and Availability Zones.

- Get information about the group and running instances.
- Enable Amazon CloudWatch metrics collection on the group.
- Update the desired capacity of the group and wait for an instance to start.
- Terminate an instance in the group.
- List scaling activities that occur in response to user requests and capacity changes.
- Get statistics for CloudWatch metrics that are collected during the example.
- Stop collecting metrics, terminate all instances, and delete the group.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
def run_scenario(as_wrapper, svc_helper):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the Amazon EC2 Auto Scaling demo for managing groups and
instances.")
    print('*'*88)

    print("This example requires a launch template that specifies how to create\n"
          "EC2 instances. You can use an existing template or create a new one.")
    template_name = q.ask(
        "Enter the name of an existing launch template or press Enter to create a new
one: ")
    template = None
    if template_name:
        template = svc_helper.get_template(template_name)
    if template is None:
        inst_type = 't1.micro'
        ami_id = 'ami-0ca285d4c2cda3300'
        print("Let's create a launch template with the following specifications:")
        print(f"\tInstanceType: {inst_type}")
        print(f"\tAMI ID: {ami_id}")
        template_name = q.ask("Enter a name for the template: ", q.non_empty)
        template = svc_helper.create_template(template_name, inst_type, ami_id)
    print('*'*88)

    print("Let's create an Auto Scaling group.")
    group_name = q.ask("Enter a name for the group: ", q.non_empty)
    zones = svc_helper.get_availability_zones()
    print("EC2 instances can be created in the following Availability Zones:")
    for index, zone in enumerate(zones):
        print(f"\t{index+1}. {zone}")
    print(f"\t{len(zones)+1}. All zones")
    zone_sel = q.ask("Which zone do you want to use? ", q.is_int, q.in_range(1,
len(zones)+1))
    group_zones = [zones[zone_sel-1]] if zone_sel <= len(zones) else zones
    print(f"Creating group {group_name}...")
    as_wrapper.create_group(group_name, group_zones, template_name, 1, 1)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    print("Created group:")
    pp(group)
    print("Waiting for instance to start...")
    wait_for_instances(group_name, as_wrapper)
```

```

print('*88)

use_metrics = q.ask(
    "Do you want to collect metrics about Amazon EC2 Auto Scaling during this demo
(y/n)? ",
    q.is_yesno)
if use_metrics:
    as_wrapper.enable_metrics(
        group_name,
        [
            'GroupMinSize', 'GroupMaxSize', 'GroupDesiredCapacity',
            'GroupInServiceInstances', 'GroupTotalInstances'])
    print(f"Metrics enabled for {group_name}.")
print('*88)

print(f"Let's update the maximum number of instances in {group_name} from 1 to 3.")
q.ask("Press Enter when you're ready.")
as_wrapper.update_group(group_name, MaxSize=3)
group = as_wrapper.describe_group(group_name)
print("The group still has one running instance, but can have up to three:")
print_simplified_group(group)
print('*88)

print(f"Let's update the desired capacity of {group_name} from 1 to 2.")
q.ask("Press Enter when you're ready.")
as_wrapper.set_desired_capacity(group_name, 2)
wait(10)
group = as_wrapper.describe_group(group_name)
print("Here's the current state of the group:")
print_simplified_group(group)
print('*88)
print("Waiting for the new instance to start...")
instance_ids = wait_for_instances(group_name, as_wrapper)
print('*88)

print(f"Let's terminate one of the instances in {group_name}.")
print("Because the desired capacity is 2, another instance will start.")
print("The currently running instances are:")
for index, inst_id in enumerate(instance_ids):
    print(f"\t{index+1}. {inst_id}")
inst_sel = q.ask(
    "Which instance do you want to stop?", q.is_int, q.in_range(1,
len(instance_ids)+1))
print(f"Stopping {instance_ids[inst_sel-1]}...")
as_wrapper.terminate_instance(instance_ids[inst_sel-1], False)
wait(10)
group = as_wrapper.describe_group(group_name)
print(f"Here's the state of {group_name}:")
print_simplified_group(group)
print("Waiting for the scaling activities to complete...")
wait_for_instances(group_name, as_wrapper)
print('*88)

print(f"Let's get a report of scaling activities for {group_name}.")
q.ask("Press Enter when you're ready.")
activities = as_wrapper.describe_scaling_activities(group_name)
print(f"Found {len(activities)} activities.\n"
      f"Activities are ordered with the most recent one first:")
for act in activities:
    pp(act)
print('*88)

if use_metrics:
    print("Let's look at CloudWatch metrics.")
    metric_namespace = 'AWS/AutoScaling'
    metric_dimensions = [{'Name': 'AutoScalingGroupName', 'Value': group_name}]
    print(f"The following metrics are enabled for {group_name}:")

```

```

done = False
while not done:
    metrics = svc_helper.get_metrics(metric_namespace, metric_dimensions)
    for index, metric in enumerate(metrics):
        print(f"\t{index+1}. {metric.name}")
    print(f"\t{len(metrics)+1}. None")
    metric_sel = q.ask(
        "Which metric do you want to see? ", q.is_int, q.in_range(1,
len(metrics)+1))
    if metric_sel < len(metrics)+1:
        span = 5
        metric = metrics[metric_sel - 1]
        print(f"Over the last {span} minutes, {metric.name} recorded:")
        # CloudWatch metric times are in the UTC+0 time zone.
        now = datetime.now(timezone.utc)
        metric_data = svc_helper.get_metric_statistics(
            metric_dimensions, metric, now-timedelta(minutes=span), now)
        pp(metric_data)
        if not q.ask("Do you want to see another metric (y/n)? ", q.is_yesno):
            done = True
    else:
        done = True

print(f"Let's clean up.")
q.ask("Press Enter when you're ready.")
if use_metrics:
    print(f"Stopping metrics collection for {group_name}.")
    as_wrapper.disable_metrics(group_name)

    print("You must terminate all instances in the group before you can delete the
group.")
    print("Set minimum size to 0.")
    as_wrapper.update_group(group_name, MinSize=0)
    group = as_wrapper.describe_group(group_name)
    instance_ids = [inst['InstanceId'] for inst in group['Instances']]
    for inst_id in instance_ids:
        print(f"Stopping {inst_id}.")
        as_wrapper.terminate_instance(inst_id, True)
    print("Waiting for instances to stop...")
    wait_for_instances(group_name, as_wrapper)
    print(f"Deleting {group_name}.")
    as_wrapper.delete_group(group_name)
    print('*'*88)

    if template is not None:
        if q.ask(f"Do you want to delete launch template {template_name} used in this
demo (y/n)? "):
            svc_helper.delete_template(template_name)
            print("Template deleted.")

    print("\nThanks for watching!")
    print('*'*88)

if __name__ == '__main__':
    try:
        wrapper = AutoScalingWrapper(boto3.client('autoscaling'))
        helper = ServiceHelper(boto3.client('ec2'), boto3.resource('cloudwatch'))
        run_scenario(wrapper, helper)
    except Exception:
        logging.exception("Something went wrong with the demo!")

```

Define functions that are called by the scenario to manage launch templates and metrics. These functions wrap Amazon EC2 and CloudWatch actions.

```
class ServiceHelper:  
    """Encapsulates Amazon EC2 and CloudWatch actions for the example."""  
    def __init__(self, ec2_client, cloudwatch_resource):  
        """  
        :param ec2_client: A Boto3 Amazon EC2 client.  
        :param cloudwatch_resource: A Boto3 CloudWatch resource.  
        """  
        self.ec2_client = ec2_client  
        self.cloudwatch_resource = cloudwatch_resource  
  
    def get_template(self, template_name):  
        """  
        Gets a launch template. Launch templates specify configuration for instances  
        that are launched by Amazon EC2 Auto Scaling.  
  
        :param template_name: The name of the template to look up.  
        :return: The template, if it exists.  
        """  
        try:  
            response =  
self.ec2_client.describe_launch_templates(LaunchTemplateNames=[template_name])  
            template = response['LaunchTemplates'][0]  
        except ClientError as err:  
            if err.response['Error']['Code'] ==  
'InvalidLaunchTemplateName.NotFoundException':  
                logger.warning("Launch template %s does not exist.", template_name)  
            else:  
                logger.error(  
                    "Couldn't verify launch template %s. Here's why: %s: %s",  
template_name,  
                    err.response['Error']['Code'], err.response['Error']['Message'])  
                raise  
        else:  
            return template  
  
    def create_template(self, template_name, inst_type, ami_id):  
        """  
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto Scaling.  
  
        :param template_name: The name to give to the template.  
        :param inst_type: The type of the instance, such as t1.micro.  
        :param ami_id: The ID of the Amazon Machine Image (AMI) to use when creating  
            an instance.  
        :return: Information about the newly created template.  
        """  
        try:  
            response = self.ec2_client.create_launch_template(  
                LaunchTemplateName=template_name,  
                LaunchTemplateData={  
                    'InstanceType': inst_type,  
                    'ImageId': ami_id})  
            template = response['LaunchTemplate']  
        except ClientError as err:  
            logger.error(  
                "Couldn't create launch template %s. Here's why: %s: %s",  
template_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return template  
  
    def delete_template(self, template_name):  
        """  
        Deletes a launch template.
```

```
:param template_name: The name of the template to delete.  
"""  
try:  
    self.ec2_client.delete_launch_template(LaunchTemplateName=template_name)  
except ClientError as err:  
    logger.error(  
        "Couldn't delete launch template %. Here's why: %s: %s",  
        template_name,  
        err.response['Error']['Code'], err.response['Error']['Message'])  
raise  
  
def get_availability_zones(self):  
    """  
    Gets a list of Availability Zones in the AWS Region of the Amazon EC2 client.  
    :return: The list of Availability Zones for the client Region.  
    """  
try:  
    response = self.ec2_client.describe_availability_zones()  
    zones = [zone['ZoneName'] for zone in response['AvailabilityZones']]  
except ClientError as err:  
    logger.error(  
        "Couldn't get availability zones. Here's why: %s: %s",  
        err.response['Error']['Code'], err.response['Error']['Message'])  
raise  
else:  
    return zones  
  
def get_metrics(self, namespace, dimensions):  
    """  
    Gets a list of CloudWatch metrics filtered by namespace and dimensions.  
    :param namespace: The namespace of the metrics to look up.  
    :param dimensions: The dimensions of the metrics to look up.  
    :return: The list of metrics.  
    """  
try:  
    metrics = list(self.cloudwatch_resource.metrics.filter(  
        Namespace=namespace, Dimensions=dimensions))  
except ClientError as err:  
    logger.error(  
        "Couldn't get metrics for %s, %s. Here's why: %s: %s", namespace,  
        dimensions,  
        err.response['Error']['Code'], err.response['Error']['Message'])  
raise  
else:  
    return metrics  
  
@staticmethod  
def get_metric_statistics(dimensions, metric, start, end):  
    """  
    Gets statistics for a CloudWatch metric within a specified time span.  
    :param dimensions: The dimensions of the metric.  
    :param metric: The metric to look up.  
    :param start: The start of the time span for retrieved metrics.  
    :param end: The end of the time span for retrieved metrics.  
    :return: The list of data points found for the specified metric.  
    """  
try:  
    response = metric.get_statistics(  
        Dimensions=dimensions, StartTime=start, EndTime=end, Period=60,  
        Statistics=['Sum'])  
    data = response['Datapoints']  
except ClientError as err:  
    logger.error(
```

```

        "Couldn't get statistics for metric %s. Here's why: %s: %s",
metric.name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return data

def print_simplified_group(group):
"""
Prints a subset of data for an Auto Scaling group.
"""
print(group['AutoScalingGroupName'])
print(f"\tLaunch template: {group['LaunchTemplate']['LaunchTemplateName']}")
print(f"\tMin: {group['MinSize']}, Max: {group['MaxSize']}, Desired:
{group['DesiredCapacity']}")
if group['Instances']:
    print(f"\tInstances:")
    for inst in group['Instances']:
        print(f"\t\t{inst['InstanceId']}: {inst['LifecycleState']}")

def wait_for_instances(group_name, as_wrapper):
"""
Waits for instances to start or stop in an Auto Scaling group.
Prints the data for each instance after scaling activities are complete.
"""
ready = False
instance_ids = []
instances = []
while not ready:
    group = as_wrapper.describe_group(group_name)
    instance_ids = [i['InstanceId'] for i in group['Instances']]
    instances = as_wrapper.describe_instances(instance_ids) if instance_ids else []
    if all([x['LifecycleState'] in ['Terminated', 'InService'] for x in
instances]):
        ready = True
    else:
        wait(10)
    if instances:
        print(f"Here are the details of the instance{'s' if len(instances) > 1 else
'}:")
        for instance in instances:
            pp(instance)
return instance_ids

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.

- [CreateAutoScalingGroup](#)
- [DeleteAutoScalingGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAutoScalingInstances](#)
- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## Amazon EMR examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon EMR.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3937\)](#)
- [Scenarios \(p. 3942\)](#)

## Actions

### Add steps to a job flow

The following code example shows how to add steps to an Amazon EMR job flow.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Add a Spark step, which is run by the cluster as soon as it is added.

```
def add_step(cluster_id, name, script_uri, script_args, emr_client):
    """
    Adds a job step to the specified cluster. This example adds a Spark
    step, which is run by the cluster as soon as it is added.

    :param cluster_id: The ID of the cluster.
    :param name: The name of the step.
    :param script_uri: The URI where the Python script is stored.
    :param script_args: Arguments to pass to the Python script.
    :param emr_client: The Boto3 EMR client object.
    :return: The ID of the newly added step.
    """
    try:
        response = emr_client.add_job_flow_steps(
            JobFlowId=cluster_id,
            Steps=[{
                'Name': name,
                'ActionOnFailure': 'CONTINUE',
                'HadoopJarStep': {
                    'Jar': 'command-runner.jar',
                    'Args': ['spark-submit', '--deploy-mode', 'cluster',
                            script_uri, *script_args]
                }
            }])
        step_id = response['StepIds'][0]
        logger.info("Started step with ID %s", step_id)
    except ClientError:
        logger.exception("Couldn't start step %s with URI %s.", name, script_uri)
        raise
    else:
        return step_id
```

Run an Amazon EMR File System (EMRFS) command as a job step on a cluster. This can be used to automate EMRFS commands on a cluster instead of running commands manually through an SSH connection.

```
import boto3
from botocore.exceptions import ClientError


def add_emrfs_step(command, bucket_url, cluster_id, emr_client):
    """
    Add an EMRFS command as a job flow step to an existing cluster.

    :param command: The EMRFS command to run.
    :param bucket_url: The URL of a bucket that contains tracking metadata.
    :param cluster_id: The ID of the cluster to update.
    :param emr_client: The Boto3 Amazon EMR client object.
    :return: The ID of the added job flow step. Status can be tracked by calling
            the emr_client.describe_step() function.
    """
    job_flow_step = {
        'Name': 'Example EMRFS Command Step',
        'ActionOnFailure': 'CONTINUE',
        'HadoopJarStep': {
            'Jar': 'command-runner.jar',
            'Args': [
                '/usr/bin/emrfs',
                command,
                bucket_url
            ]
        }
    }

    try:
        response = emr_client.add_job_flow_steps(
            JobFlowId=cluster_id, Steps=[job_flow_step])
        step_id = response['StepIds'][0]
        print(f"Added step {step_id} to cluster {cluster_id}.")
    except ClientError:
        print(f"Couldn't add a step to cluster {cluster_id}.")
        raise
    else:
        return step_id


def usage_demo():
    emr_client = boto3.client('emr')
    # Assumes the first waiting cluster has EMRFS enabled and has created metadata
    # with the default name of 'EmrFSMetadata'.
    cluster = emr_client.list_clusters(ClusterStates=['WAITING'])['Clusters'][0]
    add_emrfs_step(
        'sync', 's3://elasticmapreduce/samples/cloudfront', cluster['Id'], emr_client)

if __name__ == '__main__':
    usage_demo()
```

- For API details, see [AddJobFlowSteps](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a cluster

The following code example shows how to describe an Amazon EMR cluster.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def describe_cluster(cluster_id, emr_client):
    """
    Gets detailed information about a cluster.

    :param cluster_id: The ID of the cluster to describe.
    :param emr_client: The Boto3 EMR client object.
    :return: The retrieved cluster information.
    """
    try:
        response = emr_client.describe_cluster(ClusterId=cluster_id)
        cluster = response['Cluster']
        logger.info("Got data for cluster %s.", cluster['Name'])
    except ClientError:
        logger.exception("Couldn't get data for cluster %s.", cluster_id)
        raise
    else:
        return cluster
```

- For API details, see [DescribeCluster in AWS SDK for Python \(Boto3\) API Reference](#).

## Describe a step

The following code example shows how to describe a step on an Amazon EMR cluster.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def describe_step(cluster_id, step_id, emr_client):
    """
    Gets detailed information about the specified step, including the current state of
    the step.

    :param cluster_id: The ID of the cluster.
    :param step_id: The ID of the step.
    :param emr_client: The Boto3 EMR client object.
    :return: The retrieved information about the specified step.
    """
    try:
        response = emr_client.describe_step(ClusterId=cluster_id, StepId=step_id)
        step = response['Step']
        logger.info("Got data for step %s.", step_id)
    except ClientError:
        logger.exception("Couldn't get data for step %s.", step_id)
        raise
    else:
        return step
```

- For API details, see [DescribeStep in AWS SDK for Python \(Boto3\) API Reference](#).

## List steps for a cluster

The following code example shows how to list steps for an Amazon EMR cluster.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_steps(cluster_id, emr_client):
    """
    Gets a list of steps for the specified cluster. In this example, all steps are
    returned, including completed and failed steps.

    :param cluster_id: The ID of the cluster.
    :param emr_client: The Boto3 EMR client object.
    :return: The list of steps for the specified cluster.
    """

    try:
        response = emr_client.list_steps(ClusterId=cluster_id)
        steps = response['Steps']
        logger.info("Got %s steps for cluster %s.", len(steps), cluster_id)
    except ClientError:
        logger.exception("Couldn't get steps for cluster %s.", cluster_id)
        raise
    else:
        return steps
```

- For API details, see [ListSteps in AWS SDK for Python \(Boto3\) API Reference](#).

## Run a job flow

The following code example shows how to run an Amazon EMR job flow.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def run_job_flow(
    name, log_uri, keep_alive, applications, job_flow_role, service_role,
    security_groups, steps, emr_client):
    """
    Runs a job flow with the specified steps. A job flow creates a cluster of
    instances and adds steps to be run on the cluster. Steps added to the cluster
    are run as soon as the cluster is ready.

    This example uses the 'emr-5.30.1' release. A list of recent releases can be
    found here:
        https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-release-
    components.html.

    :param name: The name of the cluster.
    :param log_uri: The URI where logs are stored. This can be an Amazon S3 bucket URL,
        such as 's3://my-log-bucket'.
    :param keep_alive: When True, the cluster is put into a Waiting state after all
```

```

        steps are run. When False, the cluster terminates itself when
        the step queue is empty.
:param applications: The applications to install on each instance in the cluster,
                     such as Hive or Spark.
:param job_flow_role: The IAM role assumed by the cluster.
:param service_role: The IAM role assumed by the service.
:param security_groups: The security groups to assign to the cluster instances.
                        Amazon EMR adds all needed rules to these groups, so
                        they can be empty if you require only the default rules.
:param steps: The job flow steps to add to the cluster. These are run in order
              when the cluster is ready.
:param emr_client: The Boto3 EMR client object.
:returns: The ID of the newly created cluster.
"""

try:
    response = emr_client.run_job_flow(
        Name=name,
        LogUri=log_uri,
        ReleaseLabel='emr-5.30.1',
        Instances={
            'MasterInstanceType': 'm5.xlarge',
            'SlaveInstanceType': 'm5.xlarge',
            'InstanceCount': 3,
            'KeepJobFlowAliveWhenNoSteps': keep_alive,
            'EmrManagedMasterSecurityGroup': security_groups['manager'].id,
            'EmrManagedSlaveSecurityGroup': security_groups['worker'].id,
        },
        Steps=[{
            'Name': step['name'],
            'ActionOnFailure': 'CONTINUE',
            'HadoopJarStep': {
                'Jar': 'command-runner.jar',
                'Args': ['spark-submit', '--deploy-mode', 'cluster',
                         step['script_uri'], *step['script_args']]
            }
        } for step in steps],
        Applications=[{
            'Name': app
        } for app in applications],
        JobFlowRole=job_flow_role.name,
        ServiceRole=service_role.name,
        EbsRootVolumeSize=10,
        VisibleToAllUsers=True
    )
    cluster_id = response['JobFlowId']
    logger.info("Created cluster %s.", cluster_id)
except ClientError:
    logger.exception("Couldn't create cluster.")
    raise
else:
    return cluster_id

```

- For API details, see [RunJobFlow](#) in *AWS SDK for Python (Boto3) API Reference*.

## Terminate job flows

The following code example shows how to terminate Amazon EMR job flows.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def terminate_cluster(cluster_id, emr_client):
    """
    Terminates a cluster. This terminates all instances in the cluster and cannot
    be undone. Any data not saved elsewhere, such as in an Amazon S3 bucket, is lost.

    :param cluster_id: The ID of the cluster to terminate.
    :param emr_client: The Boto3 EMR client object.
    """
    try:
        emr_client.terminate_job_flows(JobFlowIds=[cluster_id])
        logger.info("Terminated cluster %s.", cluster_id)
    except ClientError:
        logger.exception("Couldn't terminate cluster %s.", cluster_id)
        raise
```

- For API details, see [TerminateJobFlows](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Run a shell script to install libraries on instances

The following code example shows how to use AWS Systems Manager to run a shell script on Amazon EMR instances to install additional libraries. This can be used to automate instance management instead of running commands manually through an SSH connection.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import argparse
import time
import boto3

def install_libraries_on_core_nodes(
    cluster_id, script_path, emr_client, ssm_client):
    """
    Copies and runs a shell script on the core nodes in the cluster.

    :param cluster_id: The ID of the cluster.
    :param script_path: The path to the script, typically an Amazon S3 object URL.
    :param emr_client: The Boto3 Amazon EMR client.
    :param ssm_client: The Boto3 AWS Systems Manager client.
    """
    core_nodes = emr_client.list_instances(
        ClusterId=cluster_id, InstanceGroupTypes=['CORE'])['Instances']
    core_instance_ids = [node['Ec2InstanceId'] for node in core_nodes]
    print(f"Found core instances: {core_instance_ids}.")

    commands = [
        # Copy the shell script from Amazon S3 to each node instance.
        f"aws s3 cp {script_path} /home/hadoop",
        # Run the shell script to install libraries on each node instance.
        "bash /home/hadoop/install_libraries.sh"]
    for command in commands:
        print(f"Sending '{command}' to core instances...")
        command_id = ssm_client.send_command(
```

```
InstanceIds=core_instance_ids,
DocumentName='AWS-RunShellScript',
Parameters={"commands": [command]},
TimeoutSeconds=3600)['Command']['CommandId']

while True:
    # Verify the previous step succeeded before running the next step.
    cmd_result = ssm_client.list_commands(
        CommandId=command_id)['Commands'][0]
    if cmd_result['StatusDetails'] == 'Success':
        print(f"Command succeeded.")
        break
    elif cmd_result['StatusDetails'] in ['Pending', 'InProgress']:
        print(f"Command status is {cmd_result['StatusDetails']}, waiting...")
        time.sleep(10)
    else:
        print(f"Command status is {cmd_result['StatusDetails']}, quitting.")
        raise RuntimeError(
            f"Command {command} failed to run. "
            f"Details: {cmd_result['StatusDetails']}")

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('cluster_id', help="The ID of the cluster.")
    parser.add_argument('script_path', help="The path to the script in Amazon S3.")
    args = parser.parse_args()

    emr_client = boto3.client('emr')
    ssm_client = boto3.client('ssm')

    install_libraries_on_core_nodes(
        args.cluster_id, args.script_path, emr_client, ssm_client)

if __name__ == '__main__':
    main()
```

- For API details, see [ListInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## AWS Glue examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with AWS Glue.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3943\)](#)
- [Scenarios \(p. 3953\)](#)

## Actions

### Create a crawler

The following code example shows how to create an AWS Glue crawler.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client

    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
        Creates a crawler that can crawl the specified target and populate a
        database in your AWS Glue Data Catalog with metadata that describes the data
        in the target.

        :param name: The name of the crawler.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and Access
                         Management (IAM) role that grants permission to let AWS Glue
                         access the resources it needs.
        :param db_name: The name to give the database that is created by the crawler.
        :param db_prefix: The prefix to give any database tables that are created by
                          the crawler.
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
                Targets={'S3Targets': [{'Path': s3_target}]})
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

- For API details, see [CreateCrawler](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a job definition

The following code example shows how to create an AWS Glue job definition.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
```

```
    self.glue_client = glue_client

def create_job(self, name, description, role_arn, script_location):
    """
    Creates a job definition for an extract, transform, and load (ETL) job that can
    be run by AWS Glue.

    :param name: The name of the job definition.
    :param description: The description of the job definition.
    :param role_arn: The ARN of an IAM role that grants AWS Glue the permissions
        it requires to run the job.
    :param script_location: The Amazon S3 URL of a Python ETL script that is run as
        part of the job. The script defines how the data is
        transformed.
    """
    try:
        self.glue_client.create_job(
            Name=name, Description=description, Role=role_arn,
            Command={'Name': 'glueetl', 'ScriptLocation': script_location,
'PythonVersion': '3'},
            GlueVersion='3.0')
    except ClientError as err:
        logger.error(
            "Couldn't create job %s. Here's why: %s: %s",
            name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
```

- For API details, see [CreateJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a crawler

The following code example shows how to delete an AWS Glue crawler.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client

    def delete_crawler(self, name):
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

- For API details, see [DeleteCrawler](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a database from the Data Catalog

The following code example shows how to delete a database from the AWS Glue Data Catalog.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def delete_database(self, name):  
        """  
        Deletes a metadata database from your Data Catalog.  
  
        :param name: The name of the database to delete.  
        """  
        try:  
            self.glue_client.delete_database(Name=name)  
        except ClientError as err:  
            logger.error(  
                "Couldn't delete database %s. Here's why: %s: %s", name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
        raise
```

- For API details, see [DeleteDatabase](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a job definition

The following code example shows how to delete an AWS Glue job definition and all associated runs.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def delete_job(self, job_name):  
        """  
        Deletes a job definition. This also deletes data about all runs that are  
        associated with this job definition.
```

```
:param job_name: The name of the job definition to delete.  
"""  
try:  
    self.glue_client.delete_job(JobName=job_name)  
except ClientError as err:  
    logger.error(  
        "Couldn't delete job %s. Here's why: %s: %s", job_name,  
        err.response['Error']['Code'], err.response['Error']['Message'])  
raise
```

- For API details, see [DeleteJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a table from a database

The following code example shows how to delete a table from an AWS Glue Data Catalog database.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def delete_table(self, db_name, table_name):  
        """  
        Deletes a table from a metadata database.  
  
        :param db_name: The name of the database that contains the table.  
        :param table_name: The name of the table to delete.  
        """  
        try:  
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)  
        except ClientError as err:  
            logger.error(  
                "Couldn't delete table %s. Here's why: %s: %s", table_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
        raise
```

- For API details, see [DeleteTable](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a crawler

The following code example shows how to get an AWS Glue crawler.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:
```

```
"""Encapsulates AWS Glue actions."""
def __init__(self, glue_client):
    """
    :param glue_client: A Boto3 Glue client.
    """
    self.glue_client = glue_client

def get_crawler(self, name):
    """
    Gets information about a crawler.

    :param name: The name of the crawler to look up.
    :return: Data about the crawler.
    """
    crawler = None
    try:
        response = self.glue_client.get_crawler(Name=name)
        crawler = response['Crawler']
    except ClientError as err:
        if err.response['Error']['Code'] == 'EntityNotFoundException':
            logger.info("Crawler %s doesn't exist.", name)
        else:
            logger.error(
                "Couldn't get crawler %s. Here's why: %s: %s",
                name,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
    return crawler
```

- For API details, see [GetCrawler](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a database from the Data Catalog

The following code example shows how to get a database from the AWS Glue Data Catalog.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client

    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
```

```
        else:
            return response['Database']
```

- For API details, see [GetDatabase](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a job run

The following code example shows how to get an AWS Glue job run.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client

    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name, run_id,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['JobRun']
```

- For API details, see [GetJobRun](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get runs of a job

The following code example shows how to get runs of an AWS Glue job.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
```

```
:param glue_client: A Boto3 Glue client.  
"""  
    self.glue_client = glue_client  
  
def get_job_runs(self, job_name):  
    """  
        Gets information about runs that have been performed for a specific job  
        definition.  
  
    :param job_name: The name of the job definition to look up.  
    :return: The list of job runs.  
    """  
    try:  
        response = self.glue_client.get_job_runs(JobName=job_name)  
    except ClientError as err:  
        logger.error(  
            "Couldn't get job runs for %s. Here's why: %s: %s", job_name,  
            err.response['Error']['Code'], err.response['Error']['Message'])  
        raise  
    else:  
        return response['JobRuns']
```

- For API details, see [GetJobRuns](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get tables from a database

The following code example shows how to get tables from a database in the AWS Glue Data Catalog.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
            :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def get_tables(self, db_name):  
        """  
            Gets a list of tables in a Data Catalog database.  
  
        :param db_name: The name of the database to query.  
        :return: The list of tables in the database.  
        """  
        try:  
            response = self.glue_client.get_tables(DatabaseName=db_name)  
        except ClientError as err:  
            logger.error(  
                "Couldn't get tables %s. Here's why: %s: %s", db_name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return response['TableList']
```

- For API details, see [GetTables](#) in *AWS SDK for Python (Boto3) API Reference*.

## List job definitions

The following code example shows how to list AWS Glue job definitions.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def list_jobs(self):  
        """  
        Lists the names of job definitions in your account.  
  
        :return: The list of job definition names.  
        """  
        try:  
            response = self.glue_client.list_jobs()  
        except ClientError as err:  
            logger.error(  
                "Couldn't list jobs. Here's why: %s: %s",  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return response['JobNames']
```

- For API details, see [ListJobs](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start a crawler

The following code example shows how to start an AWS Glue crawler.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def start_crawler(self, name):  
        """  
        Starts a crawler. The crawler crawls its configured target and creates  
        metadata that describes the data it finds in the target data source.  
        """
```

```
:param name: The name of the crawler to start.  
"""  
try:  
    self.glue_client.start_crawler(Name=name)  
except ClientError as err:  
    logger.error(  
        "Couldn't start crawler %s. Here's why: %s: %s", name,  
        err.response['Error']['Code'], err.response['Error']['Message'])  
    raise
```

- For API details, see [StartCrawler](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start a job run

The following code example shows how to start an AWS Glue job run.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def start_job_run(self, name, input_database, input_table, output_bucket_name):  
        """  
        Starts a job run. A job run extracts data from the source, transforms it,  
        and loads it to the output bucket.  
  
        :param name: The name of the job definition.  
        :param input_database: The name of the metadata database that contains tables  
            that describe the source data. This is typically created  
            by a crawler.  
        :param input_table: The name of the table in the metadata database that  
            describes the source data.  
        :param output_bucket_name: The S3 bucket where the output is written.  
        :return: The ID of the job run.  
        """  
        try:  
            # The custom Arguments that are passed to this function are used by the  
            # Python ETL script to determine the location of input and output data.  
            response = self.glue_client.start_job_run(  
                JobName=name,  
                Arguments={  
                    '--input_database': input_database,  
                    '--input_table': input_table,  
                    '--output_bucket_url': f's3://{output_bucket_name}/'})  
        except ClientError as err:  
            logger.error(  
                "Couldn't start job run %s. Here's why: %s: %s", name,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return response['JobRunId']
```

- For API details, see [StartJobRun](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Get started running crawlers and jobs

The following code example shows how to:

- Create and run a crawler that crawls a public Amazon Simple Storage Service (Amazon S3) bucket and generates a metadata database that describes the CSV-formatted data it finds.
- List information about databases and tables in your AWS Glue Data Catalog.
- Create and run a job that extracts CSV data from the source Amazon S3 bucket, transforms it by removing and renaming fields, and loads JSON-formatted output into another Amazon S3 bucket.
- List information about job runs and view some of the transformed data.
- Delete all resources created by the demo.

For more information, see [Tutorial: Getting started with AWS Glue Studio](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps AWS Glue functions used in the scenario.

```
class GlueWrapper:  
    """Encapsulates AWS Glue actions."""  
    def __init__(self, glue_client):  
        """  
        :param glue_client: A Boto3 Glue client.  
        """  
        self.glue_client = glue_client  
  
    def get_crawler(self, name):  
        """  
        Gets information about a crawler.  
  
        :param name: The name of the crawler to look up.  
        :return: Data about the crawler.  
        """  
        crawler = None  
        try:  
            response = self.glue_client.get_crawler(Name=name)  
            crawler = response['Crawler']  
        except ClientError as err:  
            if err.response['Error']['Code'] == 'EntityNotFoundException':  
                logger.info("Crawler %s doesn't exist.", name)  
            else:  
                logger.error(  
                    "Couldn't get crawler %s. Here's why: %s: %s", name,  
                    err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        return crawler  
  
    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):  
        """  
        Creates a crawler that can crawl the specified target and populate a  
        database in your AWS Glue Data Catalog with metadata that describes the data  
        in the target.  
        """
```

```
:param name: The name of the crawler.
:param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and Access Management (IAM) role that grants permission to let AWS Glue access the resources it needs.
:param db_name: The name to give the database that is created by the crawler.
:param db_prefix: The prefix to give any database tables that are created by the crawler.
:param s3_target: The URL to an S3 bucket that contains data that is the target of the crawler.
"""
try:
    self.glue_client.create_crawler(
        Name=name,
        Role=role_arn,
        DatabaseName=db_name,
        TablePrefix=db_prefix,
        Targets={'S3Targets': [{'Path': s3_target}]})
except ClientError as err:
    logger.error(
        "Couldn't create crawler. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise

def start_crawler(self, name):
"""
Starts a crawler. The crawler crawls its configured target and creates metadata that describes the data it finds in the target data source.

:param name: The name of the crawler to start.
"""
try:
    self.glue_client.start_crawler(Name=name)
except ClientError as err:
    logger.error(
        "Couldn't start crawler %s. Here's why: %s: %s", name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise

def get_database(self, name):
"""
Gets information about a database in your Data Catalog.

:param name: The name of the database to look up.
:return: Information about the database.
"""
try:
    response = self.glue_client.get_database(Name=name)
except ClientError as err:
    logger.error(
        "Couldn't get database %s. Here's why: %s: %s", name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response['Database']

def get_tables(self, db_name):
"""
Gets a list of tables in a Data Catalog database.

:param db_name: The name of the database to query.
:return: The list of tables in the database.
"""
try:
    response = self.glue_client.get_tables(DatabaseName=db_name)
except ClientError as err:
    logger.error(
```

```

        "Couldn't get tables %s. Here's why: %s: %s", db_name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response['TableList']

def create_job(self, name, description, role_arn, script_location):
    """
    Creates a job definition for an extract, transform, and load (ETL) job that can
    be run by AWS Glue.

    :param name: The name of the job definition.
    :param description: The description of the job definition.
    :param role_arn: The ARN of an IAM role that grants AWS Glue the permissions
        it requires to run the job.
    :param script_location: The Amazon S3 URL of a Python ETL script that is run as
        part of the job. The script defines how the data is
        transformed.
    """
    try:
        self.glue_client.create_job(
            Name=name, Description=description, Role=role_arn,
            Command={'Name': 'glueetl', 'ScriptLocation': script_location,
'PythonVersion': '3'},
            GlueVersion='3.0')
    except ClientError as err:
        logger.error(
            "Couldn't create job %s. Here's why: %s: %s", name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def start_job_run(self, name, input_database, input_table, output_bucket_name):
    """
    Starts a job run. A job run extracts data from the source, transforms it,
    and loads it to the output bucket.

    :param name: The name of the job definition.
    :param input_database: The name of the metadata database that contains tables
        that describe the source data. This is typically created
        by a crawler.
    :param input_table: The name of the table in the metadata database that
        describes the source data.
    :param output_bucket_name: The S3 bucket where the output is written.
    :return: The ID of the job run.
    """
    try:
        # The custom Arguments that are passed to this function are used by the
        # Python ETL script to determine the location of input and output data.
        response = self.glue_client.start_job_run(
            JobName=name,
            Arguments={
                '--input_database': input_database,
                '--input_table': input_table,
                '--output_bucket_url': f's3://{output_bucket_name}/'})
    except ClientError as err:
        logger.error(
            "Couldn't start job run %s. Here's why: %s: %s", name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response['JobRunId']

def list_jobs(self):
    """
    Lists the names of job definitions in your account.

```

```
:return: The list of job definition names.  
"""  
try:  
    response = self.glue_client.list_jobs()  
except ClientError as err:  
    logger.error(  
        "Couldn't list jobs. Here's why: %s: %s",  
        err.response['Error']['Code'], err.response['Error']['Message'])  
    raise  
else:  
    return response['JobNames']  
  
def get_job_runs(self, job_name):  
    """  
    Gets information about runs that have been performed for a specific job  
    definition.  
  
    :param job_name: The name of the job definition to look up.  
    :return: The list of job runs.  
    """  
try:  
    response = self.glue_client.get_job_runs(JobName=job_name)  
except ClientError as err:  
    logger.error(  
        "Couldn't get job runs for %s. Here's why: %s: %s", job_name,  
        err.response['Error']['Code'], err.response['Error']['Message'])  
    raise  
else:  
    return response['JobRuns']  
  
def get_job_run(self, name, run_id):  
    """  
    Gets information about a single job run.  
  
    :param name: The name of the job definition for the run.  
    :param run_id: The ID of the run.  
    :return: Information about the run.  
    """  
try:  
    response = self.glue_client.get_job_run(JobName=name, RunId=run_id)  
except ClientError as err:  
    logger.error(  
        "Couldn't get job run %s/%s. Here's why: %s: %s", name, run_id,  
        err.response['Error']['Code'], err.response['Error']['Message'])  
    raise  
else:  
    return response['JobRun']  
  
def delete_job(self, job_name):  
    """  
    Deletes a job definition. This also deletes data about all runs that are  
    associated with this job definition.  
  
    :param job_name: The name of the job definition to delete.  
    """  
try:  
    self.glue_client.delete_job(JobName=job_name)  
except ClientError as err:  
    logger.error(  
        "Couldn't delete job %s. Here's why: %s: %s", job_name,  
        err.response['Error']['Code'], err.response['Error']['Message'])  
    raise  
  
def delete_table(self, db_name, table_name):  
    """  
    Deletes a table from a metadata database.
```

```

:param db_name: The name of the database that contains the table.
:param table_name: The name of the table to delete.
"""
try:
    self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
except ClientError as err:
    logger.error(
        "Couldn't delete table %s. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise

def delete_database(self, name):
    """
    Deletes a metadata database from your Data Catalog.

    :param name: The name of the database to delete.
    """
    try:
        self.glue_client.delete_database(Name=name)
    except ClientError as err:
        logger.error(
            "Couldn't delete database %s. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def delete_crawler(self, name):
    """
    Deletes a crawler.

    :param name: The name of the crawler to delete.
    """
    try:
        self.glue_client.delete_crawler(Name=name)
    except ClientError as err:
        logger.error(
            "Couldn't delete crawler %s. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

```

Create a class that runs the scenario.

```

class GlueCrawlerJobScenario:
    """
    Encapsulates a scenario that shows how to create an AWS Glue crawler and job and
    use them to transform data from CSV to JSON format.
    """
    def __init__(self, glue_client, glue_service_role, glue_bucket):
        """
        :param glue_client: A Boto3 AWS Glue client.
        :param glue_service_role: An AWS Identity and Access Management (IAM) role
            that AWS Glue can assume to gain access to the resources it requires.
        :param glue_bucket: An S3 bucket that can hold a job script and output data
            from AWS Glue job runs.
        """
        self.glue_client = glue_client
        self.glue_service_role = glue_service_role
        self.glue_bucket = glue_bucket

    @staticmethod
    def wait(seconds, tick=12):
        """

```

```
Waits for a specified number of seconds, while also displaying an animated
spinner.

:param seconds: The number of seconds to wait.
:param tick: The number of frames per second used to animate the spinner.
"""
progress = '|/-\\'
waited = 0
while waited < seconds:
    for frame in range(tick):
        sys.stdout.write(f"\r{progress[frame % len(progress)]}")
        sys.stdout.flush()
        time.sleep(1/tick)
    waited += 1

def upload_job_script(self, job_script):
"""
Uploads a Python ETL script to an S3 bucket. The script is used by the AWS Glue
job to transform data.

:param job_script: The relative path to the job script.
"""
try:
    self.glue_bucket.upload_file(Filename=job_script, Key=job_script)
    print(f"Uploaded job script '{job_script}' to the example bucket.")
except S3UploadFailedError as err:
    logger.error("Couldn't upload job script. Here's why: %s", err)
    raise

def run(
    self, crawler_name, db_name, db_prefix, data_source, job_script, job_name):
"""
Runs the scenario. This is an interactive experience that runs at a command
prompt and asks you for input throughout.

:param crawler_name: The name of the crawler used in the scenario. If the
crawler does not exist, it is created.
:param db_name: The name to give the metadata database created by the crawler.
:param db_prefix: The prefix to give tables added to the database by the
crawler.
:param data_source: The location of the data source that is targeted by the
crawler and extracted during job runs.
:param job_script: The job script that is used to transform data during job
runs.
:param job_name: The name to give the job definition that is created during the
scenario.
"""
wrapper = GlueWrapper(self.glue_client)
print(f"Checking for crawler {crawler_name}.")
crawler = wrapper.get_crawler(crawler_name)
if crawler is None:
    print(f"Creating crawler {crawler_name}.")
    wrapper.create_crawler(
        crawler_name, self.glue_service_role.arn, db_name, db_prefix,
        data_source)
    print(f"Created crawler {crawler_name}.")
    crawler = wrapper.get_crawler(crawler_name)
pprint(crawler)
print('-'*88)

print(f"When you run the crawler, it crawls data stored in {data_source} and "
      f"creates a metadata database in the AWS Glue Data Catalog that describes"
      f"\n      the data in the data source.")
print("In this example, the source data is in CSV format.")
ready = False
```

```

while not ready:
    ready = Question.ask_question(
        "Ready to start the crawler? (y/n) ", Question.is_yesno)
    wrapper.start_crawler(crawler_name)
    print("Let's wait for the crawler to run. This typically takes a few minutes.")
    crawler_state = None
    while crawler_state != 'READY':
        self.wait(10)
        crawler = wrapper.get_crawler(crawler_name)
        crawler_state = crawler['State']
        print(f"Crawler is {crawler['State']}.")

print('-'*88)

database = wrapper.get_database(db_name)
print(f"The crawler created database {db_name}:")
pprint(database)
print(f"The database contains these tables:")
tables = wrapper.get_tables(db_name)
for index, table in enumerate(tables):
    print(f"\t{index + 1}. {table['Name']}")

table_index = Question.ask_question(
    f"Enter the number of a table to see more detail: ",
    Question.is_int, Question.in_range(1, len(tables)))
pprint(tables[table_index - 1])
print('-'*88)

print(f"Creating job definition {job_name}.")
wrapper.create_job(
    job_name, "Getting started example job.", self.glue_service_role.arn,
    f's3://{{self.glue_bucket.name}}/{job_script}')
print("Created job definition.")

print(f"When you run the job, it extracts data from {data_source}, transforms
it "
      f"by using the {job_script} script, and loads the output into "
      f"S3 bucket {self.glue_bucket.name}.")
print("In this example, the data is transformed from CSV to JSON, and only a
few "
      "fields are included in the output.")

job_run_status = None
if Question.ask_question(f"Ready to run? (y/n) ", Question.is_yesno):
    job_run_id = wrapper.start_job_run(
        job_name, db_name, tables[0]['Name'], self.glue_bucket.name)
    print(f"Job {job_name} started. Let's wait for it to run.")
    while job_run_status not in ['SUCCEEDED', 'STOPPED', 'FAILED', 'TIMEOUT']:
        self.wait(10)
        job_run = wrapper.get_job_run(job_name, job_run_id)
        job_run_status = job_run['JobRunState']
        print(f"Job {job_name}/{job_run_id} is {job_run_status}.")
    print('-'*88)

    if job_run_status == 'SUCCEEDED':
        print(f>Data from your job run is stored in your S3 bucket
'{self.glue_bucket.name}':")
        try:
            keys = [obj.key for obj in
self.glue_bucket.objects.filter(Prefix='run-')]
            for index, key in enumerate(keys):
                print(f"\t{index + 1}: {key}")
            lines = 4
            key_index = Question.ask_question(
                f"Enter the number of a block to download it and see the first
{lines} ")
                f"lines of JSON output in the block: ",
                Question.is_int, Question.in_range(1, len(keys)))
            job_data = io.BytesIO()
            self.glue_bucket.download_fileobj(keys[key_index - 1], job_data)

```

```

        job_data.seek(0)
        for _ in range(lines):
            print(job_data.readline().decode('utf-8'))
    except ClientError as err:
        logger.error(
            "Couldn't get job run data. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    print('*'*88)

    job_names = wrapper.list_jobs()
    if job_names:
        print(f"Your account has {len(job_names)} jobs defined:")
        for index, job_name in enumerate(job_names):
            print(f"\t{index + 1}. {job_name}")
        job_index = Question.ask_question(
            f"Enter a number between 1 and {len(job_names)} to see the list of runs"
        for "
            f"a job: ", Question.is_int, Question.in_range(1, len(job_names)))
        job_runs = wrapper.get_job_runs(job_names[job_index - 1])
        if job_runs:
            print(f"Found {len(job_runs)} runs for job {job_names[job_index - 1]}:")
            for index, job_run in enumerate(job_runs):
                print(
                    f"\t{index + 1}. {job_run['JobRunState']} on "
                    f"[{job_run['CompletedOn']}:%Y-%m-%d %H:%M:%S]")
            run_index = Question.ask_question(
                f"Enter a number between 1 and {len(job_runs)} to see details for a
run: ",
                Question.is_int, Question.in_range(1, len(job_runs)))
            pprint(job_runs[run_index - 1])
        else:
            print(f"No runs found for job {job_names[job_index - 1]}")
    else:
        print("Your account doesn't have any jobs defined.")
    print('*'*88)

    print(f"Let's clean up. During this example we created job definition
'{job_name}'.")
    if Question.ask_question(
        "Do you want to delete the definition and all runs? (y/n) ",
    Question.is_yesno):
        wrapper.delete_job(job_name)
        print(f"Job definition '{job_name}' deleted.")
        tables = wrapper.get_tables(db_name)
        print(f"We also created database '{db_name}' that contains these tables:")
        for table in tables:
            print(f"\t{table['Name']}")
        if Question.ask_question(
            "Do you want to delete the tables and the database? (y/n) ",
    Question.is_yesno):
            for table in tables:
                wrapper.delete_table(db_name, table['Name'])
                print(f"Deleted table {table['Name']}.")
            wrapper.delete_database(db_name)
            print(f"Deleted database {db_name}.")
        print(f"We also created crawler '{crawler_name}'.")
        if Question.ask_question(
            "Do you want to delete the crawler? (y/n) ", Question.is_yesno):
            wrapper.delete_crawler(crawler_name)
            print(f"Deleted crawler {crawler_name}.")
    print('*'*88)

def parse_args(args):

```

```

"""
Parse command line arguments.

:param args: The command line arguments.
:return: The parsed arguments.
"""
parser = argparse.ArgumentParser(
    description="Runs the AWS Glue getting started with crawlers and jobs scenario."
)
"""
Before you run this scenario, set up scaffold resources by running
"'python scaffold.py deploy'.")
parser.add_argument(
    'role_name',
    help="The name of an IAM role that AWS Glue can assume. This role must grant
access "
        "to Amazon S3 and to the permissions granted by the AWSGlueServiceRole "
        "managed policy.")
parser.add_argument(
    'bucket_name',
    help="The name of an S3 bucket that AWS Glue can access to get the job script
and "
        "put job results.")
parser.add_argument(
    '--job_script',
    default='flight_etl_job_script.py',
    help="The name of the job script file that is used in the scenario.")
return parser.parse_args(args)

def main():
    args = parse_args(sys.argv[1:])
    try:
        print('--' * 88)
        print("Welcome to the AWS Glue getting started with crawlers and jobs
scenario.")
        print('--' * 88)
        scenario = GlueCrawlerJobScenario(
            boto3.client('glue'),
            boto3.resource('iam').Role(args.role_name),
            boto3.resource('s3').Bucket(args.bucket_name))
        scenario.upload_job_script(args.job_script)
        scenario.run(
            'doc-example-crawler', 'doc-example-database', 'doc-example-',
            's3://crawler-public-us-east-1/flight/2016/csv',
            args.job_script, 'doc-example-job')
        print('--' * 88)
        print("To destroy scaffold resources, including the IAM role and S3 bucket "
            "used in this scenario, run 'python scaffold.py destroy'.")
        print("\nThanks for watching!")
        print('--' * 88)
    except Exception:
        logging.exception("Something went wrong with the example.")

```

Create an ETL script that is used by AWS Glue to extract, transform, and load data during job runs.

```

import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""

```

```

These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database      The name of a metadata database that is contained in your
                          AWS Glue Data Catalog and that contains tables that describe
                          the data to be processed.
    --input_table         The name of a table in the database that describes the data to
                          be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""

args = getResolvedOptions(sys.argv, [
    "JOB_NAME", "input_database", "input_table", "output_bucket_url"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args['input_database'],
    table_name=args['input_table'],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args['output_bucket_url'], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)

job.commit()

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.

- [CreateCrawler](#)
- [CreateJob](#)
- [DeleteCrawler](#)

- [DeleteDatabase](#)
- [DeleteJob](#)
- [DeleteTable](#)
- [GetCrawler](#)
- [GetDatabase](#)
- [GetDatabases](#)
- [GetJob](#)
- [GetJobRun](#)
- [GetJobRuns](#)
- [GetTables](#)
- [ListJobs](#)
- [StartCrawler](#)
- [StartJobRun](#)

## IAM examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with AWS Identity and Access Management.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 3963\)](#)
- [Scenarios \(p. 3982\)](#)

## Actions

### Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Attach a policy to a role using the Boto3 Policy object.

```
def attach_to_role(role_name, policy_arn):  
    """  
        Attaches a policy to a role.  
  
        :param role_name: The name of the role. **Note** this is the name, not the ARN.  
        :param policy_arn: The ARN of the policy.  
    """  
    try:  
        iam.Policy(policy_arn).attach_role(RoleName=role_name)  
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)  
    except ClientError:  
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,  
                        role_name)
```

```
raise
```

Attach a policy to a role using the Boto3 Role object.

```
def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
                         role_name)
        raise
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Attach a policy to a user

The following code example shows how to attach an IAM policy to a user.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
                         user_name)
        raise
```

- For API details, see [AttachUserPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a policy

The following code example shows how to create an IAM policy.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy
                        applies to. This ARN can contain wildcards, such as
                        'arn:aws:s3:::my-bucket/*' to allow actions on all objects
                        in the bucket named 'my-bucket'.
    :return: The newly created policy.
    """

    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": actions,
                "Resource": resource_arn
            }
        ]
    }
    try:
        policy = iam.create_policy(
            PolicyName=name, Description=description,
            PolicyDocument=json.dumps(policy_doc))
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a policy version

The following code example shows how to create an IAM policy version.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                          version for the policy. Otherwise, the default
                          is not changed.
    :return: The newly created policy version.
    """


```

```
policy_doc = {
    'Version': '2012-10-17',
    'Statement': [
        {
            'Effect': 'Allow',
            'Action': actions,
            'Resource': resource_arn
        }
    ]
}
try:
    policy = iam.Policy(policy_arn)
    policy_version = policy.create_version(
        PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default)
    logger.info(
        "Created policy version %s for policy %s.",
        policy_version.version_id, policy_version.arn)
except ClientError:
    logger.exception("Couldn't create a policy version for %s.", policy_arn)
    raise
else:
    return policy_version
```

- For API details, see [CreatePolicyVersion](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a role

The following code example shows how to create an IAM role.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Principal': {'Service': service},
                'Action': 'sts:AssumeRole'
            } for service in allowed_services
        ]
    }

    try:
        role = iam.create_role(
            RoleName=role_name,
            AssumeRolePolicyDocument=json.dumps(trust_policy))
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
```

```
    return role
```

- For API details, see [CreateRole](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a service-linked role

The following code example shows how to create an IAM service-linked role.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_service_linked_role(service_name, description):
    """
    Creates a service-linked role.

    :param service_name: The name of the service that owns the role.
    :param description: A description to give the role.
    :return: The newly created role.
    """
    try:
        response = iam.meta.client.create_service_linked_role(
            AWSServiceName=service_name, Description=description)
        role = iam.Role(response['Role']['RoleName'])
        logger.info("Created service-linked role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create service-linked role for %s.", service_name)
        raise
    else:
        return role
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a user

The following code example shows how to create an IAM user.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
```

```
        raise
    else:
        return user
```

- For API details, see [CreateUser](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an access key

The following code example shows how to create an IAM access key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name, key_pair.id)
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an alias for an account

The following code example shows how to create an alias for an IAM account.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
```

```
    logger.info("Created an alias '%s' for your account.", alias)
except ClientError:
    logger.exception("Couldn't create alias '%s' for your account.", alias)
    raise
```

- For API details, see [CreateAccountAlias](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a policy

The following code example shows how to delete an IAM policy.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

- For API details, see [DeletePolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a role

The following code example shows how to delete an IAM role.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

- For API details, see [DeleteRole](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a user

The following code example shows how to delete an IAM user.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
        raise
```

- For API details, see [DeleteUser](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an access key

The following code example shows how to delete an IAM access key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info(
            "Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an account alias

The following code example shows how to delete an IAM account alias.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise
```

- For API details, see [DeleteAccountAlias](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detach a policy from a role

The following code example shows how to detach an IAM policy from a role.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Detach a policy from a role using the Boto3 Policy object.

```
def detach_from_role(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).detach_role(RoleName=role_name)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name)
        raise
```

Detach a policy from a role using the Boto3 Role object.

```
def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
```

```
        except ClientError:
            logger.exception(
                "Couldn't detach policy %s from role %s.", policy_arn, role_name)
            raise
```

- For API details, see [DetachRolePolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detach a policy from a user

The following code example shows how to detach an IAM policy from a user.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name)
        raise
```

- For API details, see [DetachUserPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Generate a credential report

The following code example shows how to generate a credential report from IAM for the current account. After the report is generated, get it by using the GetCredentialReport action.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info("Generating credentials report for your account. "
                   "Current state is %s.", response['State'])
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your account.")
```

```
        raise
    else:
        return response
```

- For API details, see [GenerateCredentialReport in AWS SDK for Python \(Boto3\) API Reference](#).

## Get a credential report

The following code example shows how to get the most recently generated credential report from IAM.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_credential_report():
    """
    Gets the most recently generated credentials report about the current account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response['Content'])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response['Content']
```

- For API details, see [GetCredentialReport in AWS SDK for Python \(Boto3\) API Reference](#).

## Get a detailed authorization report for your account

The following code example shows how to get a detailed IAM authorization report for your account.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report, such
                           as users or roles. When not specified, all resources
                           are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
```

```
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

- For API details, see [GetAccountAuthorizationDetails](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a policy

The following code example shows how to get an IAM policy.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get('Statement', None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.", policy_arn)
        raise
    else:
        return policy_statement
```

- For API details, see [GetPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a policy version

The following code example shows how to get an IAM policy version.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
```

```
# To get an attribute of a policy, the SDK first calls get_policy.
policy_doc = policy.default_version.document
policy_statement = policy_doc.get('Statement', None)
logger.info("Got default policy doc for %s.", policy.policy_name)
logger.info(policy_doc)
except ClientError:
    logger.exception("Couldn't get default policy statement for %s.", policy_arn)
    raise
else:
    return policy_statement
```

- For API details, see [GetPolicyVersion](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a role

The following code example shows how to get an IAM role.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_role(role_name):
    """
    Gets a role by name.

    :param role_name: The name of the role to retrieve.
    :return: The specified role.
    """
    try:
        role = iam.Role(role_name)
        role.load() # calls GetRole to load attributes
        logger.info("Got role with arn %s.", role.arn)
    except ClientError:
        logger.exception("Couldn't get role named %s.", role_name)
        raise
    else:
        return role
```

- For API details, see [GetRole](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a summary of account usage

The following code example shows how to get a summary of account usage from IAM.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
```

```
try:
    summary = iam.AccountSummary()
    logger.debug(summary.summary_map)
except ClientError:
    logger.exception("Couldn't get a summary for your account.")
    raise
else:
    return summary.summary_map
```

- For API details, see [GetAccountSummary](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get data about the last use of an access key

The following code example shows how to get data about the last use of an IAM access key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response['AccessKeyLastUsed'].get('LastUsedDate', None)
        last_service = response['AccessKeyLastUsed'].get('ServiceName', None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.", key_id,
            response['UserName'], last_used_date, last_service)
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response
```

- For API details, see [GetAccessKeyLastUsed](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get the account password policy

The following code example shows how to get the IAM account password policy.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def print_password_policy():
    """
    Prints the password policy for the account.
    """
    try:
```

```
pw_policy = iam.AccountPasswordPolicy()
print("Current account password policy:")
print(f"\tallow_users_to_change_password:
{pw_policy.allow_users_to_change_password}")
print(f"\texpire_passwords: {pw_policy.expire_passwords}")
print(f"\thard_expiry: {pw_policy.hard_expiry}")
print(f"\tmax_password_age: {pw_policy.max_password_age}")
print(f"\tminimum_password_length: {pw_policy.minimum_password_length}")
print(f"\tpassword_reuse_prevention: {pw_policy.password_reuse_prevention}")
print(f"\trequire_lowercase_characters:
{pw_policy.require_lowercase_characters}")
print(f"\trequire_numbers: {pw_policy.require_numbers}")
print(f"\trequire_symbols: {pw_policy.require_symbols}")
print(f"\trequire_uppercase_characters:
{pw_policy.require_uppercase_characters}")
printed = True
except ClientError as error:
    if error.response['Error']['Code'] == 'NoSuchEntity':
        print("The account does not have a password policy set.")
    else:
        logger.exception("Couldn't get account password policy.")
        raise
else:
    return printed
```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## List SAML providers

The following code example shows how to list SAML providers for IAM.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_saml_providers(count):
    """
    Lists the SAML providers for the account.

    :param count: The maximum number of providers to list.
    """
    try:
        found = 0
        for provider in iam.saml_providers.limit(count):
            logger.info('Got SAML provider %s.', provider.arn)
            found += 1
        if found == 0:
            logger.info("Your account has no SAML providers.")
    except ClientError:
        logger.exception("Couldn't list SAML providers.")
        raise
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for Python (Boto3) API Reference*.

## List a user's access keys

The following code example shows how to list a user's IAM access keys.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys
```

- For API details, see [ListAccessKeys](#) in *AWS SDK for Python (Boto3) API Reference*.

## List account aliases

The following code example shows how to list IAM account aliases.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response['AccountAliases']
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ','.join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
    else:
        return response['AccountAliases']
```

- For API details, see [ListAccountAliases](#) in *AWS SDK for Python (Boto3) API Reference*.

## List groups

The following code example shows how to list IAM groups.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_groups(count):
    """
    Lists the specified number of groups for the account.

    :param count: The number of groups to list.
    """
    try:
        for group in iam.groups.limit(count):
            logger.info("Group: %s", group.name)
    except ClientError:
        logger.exception("Couldn't list groups for the account.")
        raise
```

- For API details, see [ListGroup](#) in *AWS SDK for Python (Boto3) API Reference*.

## List inline policies for a role

The following code example shows how to list inline policies for an IAM role.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_policies(role_name):
    """
    Lists inline policies for a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.policies.all():
            logger.info("Got inline policy %s.", policy.name)
    except ClientError:
        logger.exception("Couldn't list inline policies for %s.", role_name)
        raise
```

- For API details, see [ListRolePolicies](#) in *AWS SDK for Python (Boto3) API Reference*.

## List policies

The following code example shows how to list IAM policies.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies
```

- For API details, see [ListPolicies](#) in *AWS SDK for Python (Boto3) API Reference*.

## List policies attached to a role

The following code example shows how to list policies attached to an IAM role.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_attached_policies(role_name):
    """
    Lists policies attached to a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.attached_policies.all():
            logger.info("Got policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't list attached policies for %s.", role_name)
        raise
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for Python (Boto3) API Reference*.

## List roles

The following code example shows how to list IAM roles.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_roles(count):
    """
```

```
Lists the specified number of roles for the account.

:param count: The number of roles to list.
"""
try:
    roles = list(iam.roles.limit(count=count))
    for role in roles:
        logger.info("Role: %s", role.name)
except ClientError:
    logger.exception("Couldn't list roles for the account.")
    raise
else:
    return roles
```

- For API details, see [ListRoles](#) in *AWS SDK for Python (Boto3) API Reference*.

## List users

The following code example shows how to list IAM users.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users
```

- For API details, see [ListUsers](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update a user

The following code example shows how to update an IAM user.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    """
```

```
:param new_user_name: The new name to assign to the user.
:return: The updated user.
"""
try:
    user = iam.User(user_name)
    user.update(NewUserName=new_user_name)
    logger.info("Renamed %s to %s.", user_name, new_user_name)
except ClientError:
    logger.exception("Couldn't update name for user %s.", user_name)
    raise
return user
```

- For API details, see [UpdateUser](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update an access key

The following code example shows how to update an IAM access key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
        deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", 'Activated' if activate else 'Deactivated', key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", 'Activate' if activate else 'Deactivate', key_id)
        raise
```

- For API details, see [UpdateAccessKey](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Create a user and assume a role

The following code example shows how to:

- Create a user who has no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.

- Assume the role and list Amazon S3 buckets using temporary credentials.
- Delete the policy, role, and user.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an IAM user and a role that grants permission to list Amazon S3 buckets. The user has rights only to assume the role. After assuming the role, use temporary credentials to list buckets for the account.

```
import json
import sys
import time
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError


def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
        print('.', end='')
        sys.stdout.flush()
    print()


def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
                         that has permissions to create users, roles, and policies
                         in the account.
    :return: The newly created user, user key, and role.
    """
    try:
        user = iam_resource.create_user(UserName=f'demo-user-{uuid4()}')
        print(f"Created user {user.name}.")
    except ClientError as error:
        print(f"Couldn't create a user for the demo. Here's why: "
              f"{error.response['Error']['Message']}")
        raise

    try:
        user_key = user.create_access_key_pair()
        print(f"Created access key pair for user {user.name}.")
    except ClientError as error:
        print(f"Couldn't create access keys for user {user.name}. Here's why: "
              f"{error.response['Error']['Message']}")
        raise

    print(f"Wait for user to be ready.", end=' ')
    progress_bar(10)
```

```

try:
    role = iam_resource.create_role(
        RoleName=f'demo-role-{uuid4()}',
        AssumeRolePolicyDocument=json.dumps({
            'Version': '2012-10-17',
            'Statement': [{
                'Effect': 'Allow',
                'Principal': {'AWS': user.arn},
                'Action': 'sts:AssumeRole'}}]))
    print(f"Created role {role.name}.")
except ClientError as error:
    print(f"Couldn't create a role for the demo. Here's why: "
          f"{error.response['Error']['Message']}")
    raise

try:
    policy = iam_resource.create_policy(
        PolicyName=f'demo-policy-{uuid4()}',
        PolicyDocument=json.dumps({
            'Version': '2012-10-17',
            'Statement': [{
                'Effect': 'Allow',
                'Action': 's3>ListAllMyBuckets',
                'Resource': 'arn:aws:s3:::*'}})))
    role.attach_policy(PolicyArn=policy.arn)
    print(f"Created policy {policy.policy_name} and attached it to the role.")
except ClientError as error:
    print(f"Couldn't create a policy and attach it to role {role.name}. Here's why:
"
          f"{error.response['Error']['Message']}")
    raise

try:
    user.create_policy(
        PolicyName=f'demo-user-policy-{uuid4()}',
        PolicyDocument=json.dumps({
            'Version': '2012-10-17',
            'Statement': [{
                'Effect': 'Allow',
                'Action': 'sts:AssumeRole',
                'Resource': role.arn]})))
    print(f"Created an inline policy for {user.name} that lets the user assume "
          f"the role.")
except ClientError as error:
    print(f"Couldn't create an inline policy for user {user.name}. Here's why: "
          f"{error.response['Error']['Message']}")
    raise

print("Give AWS time to propagate these new resources and connections.", end='')
progress_bar(10)

return user, user_key, role

def show_access_denied_without_role(user_key):
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
                     have permission to list buckets in the account.
    """
    print(f"Try to list buckets without first assuming the role.")
    s3_denied_resource = boto3.resource(
        's3', aws_access_key_id=user_key.id, aws_secret_access_key=user_key.secret)
    try:

```

```
        for bucket in s3_denied_resource.buckets.all():
            print(bucket.name)
        raise RuntimeError("Expected to get AccessDenied error when listing buckets!")
    except ClientError as error:
        if error.response['Error']['Code'] == 'AccessDenied':
            print("Attempt to list buckets with no permissions: AccessDenied.")
        else:
            raise

def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
    Assumes a role that grants permission to list the Amazon S3 buckets in the account.
    Uses the temporary credentials from the role to list the buckets that are owned
    by the assumed role's account.

    :param user_key: The access key of a user that has permission to assume the role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                           grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
        'sts', aws_access_key_id=user_key.id, aws_secret_access_key=user_key.secret)
    try:
        response = sts_clientassume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name)
        temp_credentials = response['Credentials']
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:
        print(f"Couldn't assume role {assume_role_arn}. Here's why: "
              f"{error.response['Error']['Message']}")
        raise

    # Create an S3 resource that can access the account with the temporary credentials.
    s3_resource = boto3.resource(
        's3',
        aws_access_key_id=temp_credentials['AccessKeyId'],
        aws_secret_access_key=temp_credentials['SecretAccessKey'],
        aws_session_token=temp_credentials['SessionToken'])
    print(f"Listing buckets for the assumed role's account:")
    try:
        for bucket in s3_resource.buckets.all():
            print(bucket.name)
    except ClientError as error:
        print(f"Couldn't list buckets for the account. Here's why: "
              f"{error.response['Error']['Message']}")
        raise

def teardown(user, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print("Couldn't detach policy, delete policy, or delete role. Here's why: "
```

```
        f"[error.response['Error']['Message']]")
    raise

    try:
        for user_pol in user.policies.all():
            user_pol.delete()
            print("Deleted inline user policy.")
        for key in user.access_keys.all():
            key.delete()
            print("Deleted user's access key.")
        user.delete()
        print(f"Deleted {user.name}.")
    except ClientError as error:
        print("Couldn't delete user policy or delete user. Here's why: "
              f"[error.response['Error']['Message']]")

def usage_demo():
    """Drives the demonstration."""
    print('*'*88)
    print(f"Welcome to the IAM create user and assume role demo.")
    print('*'*88)
    iam_resource = boto3.resource('iam')
    user = None
    role = None
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn, 'AssumeRoleDemoSession')
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
            teardown(user, role)
    print("Thanks for watching!")

if __name__ == '__main__':
    usage_demo()
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## Create read-only and read-write users

The following code example shows how to:

- Create two IAM users.
- Attach a policy that lets one of the users get and put objects in an Amazon Simple Storage Service (Amazon S3) bucket.
- Attach a policy that lets the other user get objects from the bucket.
- Obtain differing permissions to the bucket based on user credentials.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap IAM user actions.

```
import logging
import time

import boto3
from botocore.exceptions import ClientError

import access_key_wrapper
import policy_wrapper

logger = logging.getLogger(__name__)
iam = boto3.resource('iam')

def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user

def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user

def list_users():
    """
    Lists the users in the current account.
    
```

```

    :return: The list of users.
    """
try:
    users = list(iam.users.all())
    logger.info("Got %s users.", len(users))
except ClientError:
    logger.exception("Couldn't get users.")
    raise
else:
    return users

def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
try:
    iam.User(user_name).delete()
    logger.info("Deleted user %s.", user_name)
except ClientError:
    logger.exception("Couldn't delete user %s.", user_name)
    raise

def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
try:
    iam.User(user_name).attach_policy(PolicyArn=policy_arn)
    logger.info("Attached policy %s to user %s.", policy_arn, user_name)
except ClientError:
    logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
user_name)
    raise

def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
try:
    iam.User(user_name).detach_policy(PolicyArn=policy_arn)
    logger.info("Detached policy %s from user %s.", policy_arn, user_name)
except ClientError:
    logger.exception(
        "Couldn't detach policy %s from user %s.", policy_arn, user_name)
    raise

```

Create functions that wrap IAM policy actions.

```

import json
import logging
import operator
import pprint
import time

import boto3

```

```

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource('iam')

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
        form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy
        applies to. This ARN can contain wildcards, such as
        'arn:aws:s3:::my-bucket/*' to allow actions on all objects
        in the bucket named 'my-bucket'.
    :return: The newly created policy.
    """

    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": actions,
                "Resource": resource_arn
            }
        ]
    }
    try:
        policy = iam.create_policy(
            PolicyName=name, Description=description,
            PolicyDocument=json.dumps(policy_doc))
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy

def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """

    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise

```

Create functions that wrap IAM access key actions.

```

import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource('iam')

def create_key(user_name):

```

```

"""
Creates an access key for the specified user. Each user can have a
maximum of two keys.

:param user_name: The name of the user.
:return: The created access key.
"""

try:
    key_pair = iam.User(user_name).create_access_key_pair()
    logger.info(
        "Created access key pair for %s. Key ID is %s.",
        key_pair.user_name, key_pair.id)
except ClientError:
    logger.exception("Couldn't create access key pair for %s.", user_name)
    raise
else:
    return key_pair

def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info(
            "Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise

```

Use the wrapper functions to create users with differing policies and use their credentials to access an Amazon S3 bucket.

```

def usage_demo():
    """
    Shows how to manage users, keys, and policies.
    This demonstration creates two users: one user who can put and get objects in an
    Amazon S3 bucket, and another user who can only get objects from the bucket.
    The demo then shows how the users can perform only the actions they are permitted
    to perform.
    """

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    print('*'*88)
    print("Welcome to the AWS Identity and Account Management user demo.")
    print('*'*88)
    print("Users can have policies and roles attached to grant them specific "
          "permissions.")
    s3 = boto3.resource('s3')
    bucket = s3.create_bucket(
        Bucket=f'demo-iam-bucket-{time.time_ns()}',
        CreateBucketConfiguration={
            'LocationConstraint': s3.meta.client.meta.region_name
        }
    )
    print(f"Created an Amazon S3 bucket named {bucket.name}.")
    user_read_writer = create_user('demo-iam-read-writer')
    user_reader = create_user('demo-iam-reader')
    print(f"Created two IAM users: {user_read_writer.name} and {user_reader.name}")

```

```
update_user(user_read_writer.name, 'demo-iam-creator')
update_user(user_reader.name, 'demo-iam-getter')
users = list_users()
user_read_writer = next(user for user in users if user.user_id ==
user_read_writer.user_id)
user_reader = next(user for user in users if user.user_id == user_reader.user_id)
print(f"Changed the names of the users to {user_read_writer.name} "
      f"and {user_reader.name}.")

read_write_policy = policy_wrapper.create_policy(
    'demo-iam-read-write-policy',
    'Grants rights to create and get an object in the demo bucket.',
    ['s3:PutObject', 's3:GetObject'],
    f'arn:aws:s3::{bucket.name}/*')
print(f"Created policy {read_write_policy.policy_name} with ARN:
{read_write_policy.arn}")
print(read_write_policy.description)
read_policy = policy_wrapper.create_policy(
    'demo-iam-read-policy',
    'Grants rights to get an object from the demo bucket.',
    's3:GetObject',
    f'arn:aws:s3::{bucket.name}/*')
print(f"Created policy {read_policy.policy_name} with ARN: {read_policy.arn}")
print(read_policy.description)
attach_policy(user_read_writer.name, read_write_policy.arn)
print(f"Attached {read_write_policy.policy_name} to {user_read_writer.name}.")
attach_policy(user_reader.name, read_policy.arn)
print(f"Attached {read_policy.policy_name} to {user_reader.name}.")

user_read_writer_key = access_key_wrapper.create_key(user_read_writer.name)
print(f"Created access key pair for {user_read_writer.name}.")
user_reader_key = access_key_wrapper.create_key(user_reader.name)
print(f"Created access key pair for {user_reader.name}.")

s3_read_writer_resource = boto3.resource(
    's3',
    aws_access_key_id=user_read_writer_key.id,
    aws_secret_access_key=user_read_writer_key.secret)
demo_object_key = f'object-{time.time_ns()}''
demo_object = None
while demo_object is None:
    try:
        demo_object = s3_read_writer_resource.Bucket(bucket.name).put_object(
            Key=demo_object_key, Body=b'AWS IAM demo object content!')
    except ClientError as error:
        if error.response['Error']['Code'] == 'InvalidAccessKeyId':
            print("Access key not yet available. Waiting...")
            time.sleep(1)
        else:
            raise
print(f"Put {demo_object_key} into {bucket.name} using "
      f"{user_read_writer.name}'s credentials.")

read_writer_object = s3_read_writer_resource.Bucket(
    bucket.name).Object(demo_object_key)
read_writer_content = read_writer_object.get()['Body'].read()
print(f"Got object {read_writer_object.key} using read-writer user's credentials.")
print(f"Object content: {read_writer_content}")

s3_reader_resource = boto3.resource(
    's3',
    aws_access_key_id=user_reader_key.id,
    aws_secret_access_key=user_reader_key.secret)
demo_content = None
while demo_content is None:
    try:
```

```
    demo_object =
s3_reader_resource.Bucket(bucket.name).Object(demo_object_key)
    demo_content = demo_object.get()['Body'].read()
    print(f"Got object {demo_object.key} using reader user's credentials.")
    print(f"Object content: {demo_content}")
except ClientError as error:
    if error.response['Error']['Code'] == 'InvalidAccessKeyId':
        print("Access key not yet available. Waiting...")
        time.sleep(1)
    else:
        raise

try:
    demo_object.delete()
except ClientError as error:
    if error.response['Error']['Code'] == 'AccessDenied':
        print('*'*88)
        print("Tried to delete the object using the reader user's credentials. "
              "Got expected AccessDenied error because the reader is not "
              "allowed to delete objects.")
        print('*'*88)

access_key_wrapper.delete_key(user_reader.name, user_reader_key.id)
detach_policy(user_reader.name, read_policy.arn)
policy_wrapper.delete_policy(read_policy.arn)
delete_user(user_reader.name)
print(f"Deleted keys, detached and deleted policy, and deleted
{user_reader.name}.")

access_key_wrapper.delete_key(user_read_writer.name, user_read_writer_key.id)
detach_policy(user_read_writer.name, read_write_policy.arn)
policy_wrapper.delete_policy(read_write_policy.arn)
delete_user(user_read_writer.name)
print(f"Deleted keys, detached and deleted policy, and deleted
{user_read_writer.name}.")

bucket.objects.delete()
bucket.delete()
print(f"Emptied and deleted {bucket.name}.")
print("Thanks for watching!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [AttachUserPolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteUser](#)
  - [DetachUserPolicy](#)
  - [ListUsers](#)
  - [UpdateUser](#)

## Manage access keys

The following code example shows how to:

- Create and list access keys.

- Find out when and how an access key was last used.
- Update and delete access keys.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap IAM access key actions.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource('iam')

def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys

def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name, key_pair.id)
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair

def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response['AccessKeyLastUsed'].get('LastUsedDate', None)
    except ClientError:
        logger.exception("Couldn't get information about the key's last use.")
```

```

last_service = response['AccessKeyLastUsed'].get('ServiceName', None)
logger.info(
    "Key %s was last used by %s on %s to access %s.", key_id,
    response['UserName'], last_used_date, last_service)
except ClientError:
    logger.exception("Couldn't get last use of key %s.", key_id)
    raise
else:
    return response

def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
    deactivated.
    """
    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", 'Activated' if activate else 'Deactivated', key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", 'Activate' if activate else 'Deactivate', key_id)
        raise

def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """
    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info(
            "Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise

```

Use the wrapper functions to perform access key actions for the current user.

```

def usage_demo():
    """Shows how to create and manage access keys."""
    def print_keys():
        """Gets and prints the current keys for a user."""
        current_keys = list_keys(current_user_name)
        print("The current user's keys are now:")
        print(*[f"{key.id}: {key.status}" for key in current_keys], sep='\n')

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    print('*'*88)
    print("Welcome to the AWS Identity and Account Management access key demo.")
    print('*'*88)
    current_user_name = iam.CurrentUser().user_name

```

```
print(f"This demo creates an access key for the current user "
      f"({{current_user_name}}), manipulates the key in a few ways, and then "
      f"deletes it.")
all_keys = list_keys(current_user_name)
if len(all_keys) == 2:
    print("The current user already has the maximum of 2 access keys. To run "
          "this demo, either delete one of the access keys or use a user "
          "that has only 1 access key.")
else:
    new_key = create_key(current_user_name)
    print(f"Created a new key with id {new_key.id} and secret {new_key.secret}.")
    print_keys()
    existing_key = next(key for key in all_keys if key != new_key)
    last_use = get_last_use(existing_key.id)['AccessKeyLastUsed']
    print(f"Key {all_keys[0].id} was last used to access {last_use['ServiceName']}")

    f"on {last_use['LastUsedDate']}")
update_key(current_user_name, new_key.id, False)
print(f"Key {new_key.id} is now deactivated.")
print_keys()
delete_key(current_user_name, new_key.id)
print_keys()
print("Thanks for watching!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateAccessKey](#)
  - [DeleteAccessKey](#)
  - [GetAccessKeyLastUsed](#)
  - [ListAccessKeys](#)
  - [UpdateAccessKey](#)

## Manage policies

The following code example shows how to:

- Create and list policies.
- Create and get policy versions.
- Roll back a policy to a previous version.
- Delete policies.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap IAM policy actions.

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError
```

```

logger = logging.getLogger(__name__)
iam = boto3.resource('iam')

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy
                        applies to. This ARN can contain wildcards, such as
                        'arn:aws:s3:::my-bucket/*' to allow actions on all objects
                        in the bucket named 'my-bucket'.
    :return: The newly created policy.
    """

    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": actions,
                "Resource": resource_arn
            }
        ]
    }
    try:
        policy = iam.create_policy(
            PolicyName=name, Description=description,
            PolicyDocument=json.dumps(policy_doc))
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy

def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are returned.
    :return: The list of policies.
    """

    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies

def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                          version for the policy. Otherwise, the default
                          is not changed.
    """

```

```
:return: The newly created policy version.  
"""  
policy_doc = {  
    'Version': '2012-10-17',  
    'Statement': [  
        {  
            'Effect': 'Allow',  
            'Action': actions,  
            'Resource': resource_arn  
        }  
    ]  
}  
try:  
    policy = iam.Policy(policy_arn)  
    policy_version = policy.create_version(  
        PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default)  
    logger.info(  
        "Created policy version %s for policy %s.",  
        policy_version.version_id, policy_version.arn)  
except ClientError:  
    logger.exception("Couldn't create a policy version for %s.", policy_arn)  
    raise  
else:  
    return policy_version  
  
def get_default_policy_statement(policy_arn):  
    """  
    Gets the statement of the default version of the specified policy.  
  
    :param policy_arn: The ARN of the policy to look up.  
    :return: The statement of the default policy version.  
    """  
    try:  
        policy = iam.Policy(policy_arn)  
        # To get an attribute of a policy, the SDK first calls get_policy.  
        policy_doc = policy.default_version.document  
        policy_statement = policy_doc.get('Statement', None)  
        logger.info("Got default policy doc for %s.", policy.policy_name)  
        logger.info(policy_doc)  
    except ClientError:  
        logger.exception("Couldn't get default policy statement for %s.", policy_arn)  
        raise  
    else:  
        return policy_statement  
  
def rollback_policy_version(policy_arn):  
    """  
    Rolls back to the previous default policy, if it exists.  
  
    1. Gets the list of policy versions in order by date.  
    2. Finds the default.  
    3. Makes the previous policy the default.  
    4. Deletes the old default version.  
  
    :param policy_arn: The ARN of the policy to roll back.  
    :return: The default version of the policy after the rollback.  
    """  
    try:  
        policy_versions = sorted(  
            iam.Policy(policy_arn).versions.all(),  
            key=operator.attrgetter('create_date'))  
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)  
    except ClientError:  
        logger.exception("Couldn't get versions for %s.", policy_arn)  
        raise
```

```

default_version = None
rollback_version = None
try:
    while default_version is None:
        ver = policy_versions.pop()
        if ver.is_default_version:
            default_version = ver
    rollback_version = policy_versions.pop()
    rollback_version.set_as_default()
    logger.info("Set %s as the default version.", rollback_version.version_id)
    default_version.delete()
    logger.info("Deleted original default version %s.", default_version.version_id)
except IndexError:
    if default_version is None:
        logger.warning("No default version found for %s.", policy_arn)
    elif rollback_version is None:
        logger.warning(
            "Default version %s found for %s, but no previous version exists, so "
            "nothing to roll back to.", default_version.version_id, policy_arn)
except ClientError:
    logger.exception("Couldn't roll back version for %s.", policy_arn)
    raise
else:
    return rollback_version

def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise

```

Use the wrapper functions to create policies, update versions, and get information about them.

```

def usage_demo():
    """Shows how to use the policy functions."""
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    print('-'*88)
    print("Welcome to the AWS Identity and Account Management policy demo.")
    print('-'*88)
    print("Policies let you define sets of permissions that can be attached to "
          "other IAM resources, like users and roles.")
    bucket_arn = f'arn:aws:s3:::made-up-bucket-name'
    policy = create_policy(
        'demo-iam-policy', 'Policy for IAM demonstration.',
        ['s3>ListObjects'], bucket_arn)
    print(f"Created policy {policy.policy_name}.")
    policies = list_policies('Local')
    print(f"Your account has {len(policies)} managed policies:")
    print(*[pol.policy_name for pol in policies], sep=', ')
    time.sleep(1)
    policy_version = create_policy_version(
        policy.arn, ['s3:PutObject'], bucket_arn, True)
    print(f"Added policy version {policy_version.version_id} to policy "
          f"{policy.policy_name}.")
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is:")
    pprint.pprint(default_statement)

```

```
rollback_version = rollback_policy_version(policy.arn)
print(f"Rolled back to version {rollback_version.version_id} for "
      f"{policy.policy_name}.")
default_statement = get_default_policy_statement(policy.arn)
print(f"The default policy statement for {policy.policy_name} is now:")
pprint.pprint(default_statement)
delete_policy(policy.arn)
print(f"Deleted policy {policy.policy_name}.")
print("Thanks for watching!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreatePolicy](#)
  - [CreatePolicyVersion](#)
  - [DeletePolicy](#)
  - [DeletePolicyVersion](#)
  - [GetPolicyVersion](#)
  - [ListPolicies](#)
  - [ListPolicyVersions](#)
  - [SetDefaultPolicyVersion](#)

## Manage roles

The following code example shows how to:

- Create an IAM role.
- Attach and detach policies for a role.
- Delete a role.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap IAM role actions.

```
import json
import logging
import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource('iam')

def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        'Version': '2012-10-17',
        'Statement': [{
```

```
        'Effect': 'Allow',
        'Principal': {'Service': service},
        'Action': 'sts:AssumeRole'
    } for service in allowed_services
]
}

try:
    role = iam.create_role(
        RoleName=role_name,
        AssumeRolePolicyDocument=json.dumps(trust_policy))
    logger.info("Created role %s.", role.name)
except ClientError:
    logger.exception("Couldn't create role %s.", role_name)
    raise
else:
    return role

def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
                         role_name)
        raise

def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name)
        raise

def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

Use the wrapper functions to create a role, then attach and detach a policy.

```
def usage_demo():
```

```
"""Shows how to use the role functions."""
logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
print('*'*88)
print("Welcome to the AWS Identity and Account Management role demo.")
print('*'*88)
print("Roles let you define sets of permissions and can be assumed by "
      "other entities, like users and services.")
print("The first 10 roles currently in your account are:")
roles = list_roles(10)
print(f"The inline policies for role {roles[0].name} are:")
list_policies(roles[0].name)
role = create_role(
    'demo-iam-role',
    ['lambda.amazonaws.com', 'batchoperations.s3.amazonaws.com'])
print(f"Created role {role.name}, with trust policy:")
pprint.pprint(role.assume_role_policy_document)
policy_arn = 'arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess'
attach_policy(role.name, policy_arn)
print(f"Attached policy {policy_arn} to {role.name}.")
print(f"Policies attached to role {role.name} are:")
list_attached_policies(role.name)
detach_policy(role.name, policy_arn)
print(f"Detached policy {policy_arn} from {role.name}.")
delete_role(role.name)
print(f"Deleted {role.name}.")
print("Thanks for watching!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [AttachRolePolicy](#)
  - [CreateRole](#)
  - [DeleteRole](#)
  - [DetachRolePolicy](#)

## Manage your account

The following code example shows how to:

- Get and update the alias for the account.
- Generate a report of users and their credentials.
- Get a summary of account usage.
- Get details about all users, groups, roles, and policies in your account, including their relationships to each other.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap IAM account actions.

```
import logging
import pprint
import sys
import time
import boto3
from botocore.exceptions import ClientError
```

```
logger = logging.getLogger(__name__)
iam = boto3.resource('iam')

def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response['AccountAliases']
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ','.join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response['AccountAliases']

def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """
    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise

def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise

def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info("Generating credentials report for your account. "
                   "Current state is %s.", response['State'])
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your account.")
        raise
    else:
```

```

        return response

def get_credential_report():
    """
    Gets the most recently generated credentials report about the current account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response['Content'])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response['Content']

def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map

def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report, such
                           as users or roles. When not specified, all resources
                           are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details

```

Call wrapper functions to change the account alias and to get reports about the account.

```

def usage_demo():
    """Shows how to use the account functions."""
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    print('*'*88)
    print("Welcome to the AWS Identity and Account Management account demo.")
    print('*'*88)
    print("Setting an account alias lets you use the alias in your sign-in URL "
          "instead of your account number.")
    old_aliases = list_aliases()
    if len(old_aliases) > 0:

```

```
        print(f"Your account currently uses '{old_aliases[0]}' as its alias.")
    else:
        print("Your account currently has no alias.")
    for index in range(1, 3):
        new_alias = f'alias-{index}-{time.time_ns()}'  

        print(f"Setting your account alias to {new_alias}")
        create_alias(new_alias)
    current_aliases = list_aliases()
    print(f"Your account alias is now {current_aliases}.")
    delete_alias(current_aliases[0])
    print(f"Your account now has no alias.")
    if len(old_aliases) > 0:
        print(f"Restoring your original alias back to {old_aliases[0]}...")
        create_alias(old_aliases[0])

    print('*'*88)
    print("You can get various reports about your account.")
    print("Let's generate a credentials report...")
    report_state = None
    while report_state != 'COMPLETE':
        cred_report_response = generate_credential_report()
        old_report_state = report_state
        report_state = cred_report_response['State']
        if report_state != old_report_state:
            print(report_state, sep='')
        else:
            print('.', sep='')
        sys.stdout.flush()
        time.sleep(1)
    print()
    cred_report = get_credential_report()
    col_count = 3
    print(f"Got credentials report. Showing only the first {col_count} columns.")
    cred_lines = [line.split(',')[:col_count] for line
                 in cred_report.decode('utf-8').split('\n')]
    col_width = max([len(item) for line in cred_lines for item in line]) + 2
    for line in cred_report.decode('utf-8').split('\n'):
        print('.'.join(element.ljust(col_width)
                       for element in line.split(',')[:col_count]))

    print('*'*88)
    print("Let's get an account summary.")
    summary = get_summary()
    print("Here's your summary:")
    pprint.pprint(summary)

    print('*'*88)
    print("Let's get authorization details!")
    details = get_authorization_details([])
    see_details = input("These are pretty long, do you want to see them (y/n)? ")
    if see_details.lower() == 'y':
        pprint.pprint(details)

    print('*'*88)
    pw_policy_created = None
    see_pw_policy = input("Want to see the password policy for the account (y/n)? ")
    if see_pw_policy.lower() == 'y':
        while True:
            if print_password_policy():
                break
            else:
                answer = input("Do you want to create a default password policy (y/n)? ")
        if answer.lower() == 'y':
            pw_policy_created = iam.create_account_password_policy()
        else:
```

```
        break
    if pw_policy_created is not None:
        answer = input("Do you want to delete the password policy (y/n)? ")
        if answer.lower() == 'y':
            pw_policy_created.delete()
            print("Password policy deleted.")

    print("The SAML providers for your account are:")
    list_saml_providers(10)

    print('*'*88)
    print("Thanks for watching.")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateAccountAlias](#)
  - [DeleteAccountAlias](#)
  - [GenerateCredentialReport](#)
  - [GetAccountAuthorizationDetails](#)
  - [GetAccountSummary](#)
  - [GetCredentialReport](#)
  - [ListAccountAliases](#)

### [Roll back a policy version](#)

The following code example shows how to:

- Get the list of policy versions in order by date.
- Find the default policy version.
- Make the previous policy version the default.
- Delete the old default version.

### **SDK for Python (Boto3)**

#### **Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter('create_date'))
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
```

```
raise

default_version = None
rollback_version = None
try:
    while default_version is None:
        ver = policy_versions.pop()
        if ver.is_default_version:
            default_version = ver
    rollback_version = policy_versions.pop()
    rollback_version.set_as_default()
    logger.info("Set %s as the default version.", rollback_version.version_id)
    default_version.delete()
    logger.info("Deleted original default version %s.", default_version.version_id)
except IndexError:
    if default_version is None:
        logger.warning("No default version found for %s.", policy_arn)
    elif rollback_version is None:
        logger.warning(
            "Default version %s found for %s, but no previous version exists, so "
            "nothing to roll back to.", default_version.version_id, policy_arn)
except ClientError:
    logger.exception("Couldn't roll back version for %s.", policy_arn)
    raise
else:
    return rollback_version
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [DeletePolicyVersion](#)
  - [ListPolicyVersions](#)
  - [SetDefaultPolicyVersion](#)

## AWS KMS examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with AWS Key Management Service.

**Actions** are code excerpts that show you how to call individual service functions.

**Scenarios** are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4006\)](#)
- [Scenarios \(p. 4020\)](#)

## Actions

### Create a grant for a key

The following code example shows how to create a grant for a KMS key.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

This example grants permission to let a user generate a data key that can be used for encryption.

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    def create_grant(self, key_id):
        """
        Creates a grant for a key that lets a user generate a symmetric data
        encryption key.

        :param key_id: The ARN or ID of the key.
        :return: The grant that is created.
        """

        user = input(
            f"Enter the ARN of an IAM user to grant that user GenerateDataKey "
            f"permissions on key {key_id}.\n")
        if user != '':
            try:
                grant = self.kms_client.create_grant(
                    KeyId=key_id, GranteePrincipal=user,
                    Operations=['GenerateDataKey'])
            except ClientError as err:
                logger.error(
                    "Couldn't create a grant on key %s. Here's why: %s",
                    key_id, err.response['Error']['Message'])
            else:
                print(f"Grant created on key {key_id}.")
                return grant
        else:
            print("Skipping grant creation.)
```

- For API details, see [CreateGrant](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a key

The following code example shows how to create an AWS KMS key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    def create_key(self):
        """
        Creates a key (or multiple keys) with a user-provided description.
        """

        answer = 'y'
        while answer.lower() == 'y':
            key_desc = input("\nLet's create a key. Describe it for me: ")
            if not key_desc:
                key_desc = "Key management demo key"
            try:
                key = self.kms_client.create_key(Description=key_desc)['KeyMetadata']
            except ClientError as err:
```

```
        logging.error(
            "Couldn't create your key. Here's why: %s",
            err.response['Error']['Message'])
        raise
    else:
        print("Key created:")
        pprint(key)
        self.created_keys.append(key)
    answer = input("Create another (y/n)? ")
```

- For API details, see [CreateKey](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an alias for a key

The following code example shows how to create an alias for a KMS key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    def create_alias(self, key_id):
        """
        Creates an alias for the specified key.

        :param key_id: The ARN or ID of a key to give an alias.
        :return: The alias given to the key.
        """
        alias = ''
        while alias == '':
            alias = input(f"What alias would you like to give to key {key_id}? ")
        try:
            self.kms_client.create_alias(AliasName=alias, TargetKeyId=key_id)
        except ClientError as err:
            logger.error(
                "Couldn't create alias %s. Here's why: %s",
                alias, err.response['Error']['Message'])
        else:
            print(f"Created alias {alias} for key {key_id}.")
            return alias
```

- For API details, see [CreateAlias](#) in *AWS SDK for Python (Boto3) API Reference*.

## Decrypt ciphertext

The following code example shows how to decrypt ciphertext that was encrypted by a KMS key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    def decrypt(self, key_id, cipher_text):
        """
        Decrypts text previously encrypted with a key.

        :param key_id: The ARN or ID of the key used to decrypt the data.
        :param cipher_text: The encrypted text to decrypt.
        """
        answer = input("Ready to decrypt your ciphertext (y/n)? ")
        if answer.lower() == 'y':
            try:
                text = self.kms_client.decrypt(
                    KeyId=key_id, CiphertextBlob=cipher_text)['Plaintext']
            except ClientError as err:
                logger.error(
                    "Couldn't decrypt your ciphertext. Here's why: %s",
                    err.response['Error']['Message'])
            else:
                print(f"Your plaintext is {text.decode()}")
        else:
            print("Skipping decryption demo.")
```

- For API details, see [Decrypt in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete an alias

The following code example shows how to delete an AWS KMS alias.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    def delete_alias(self):
        """
        Deletes an alias.

        """
        alias = input(f"Enter an alias that you'd like to delete: ")
        if alias != '':
            try:
                self.kms_client.delete_alias(AliasName=alias)
            except ClientError as err:
                logger.error(
                    "Couldn't delete alias %s. Here's why: %s",
                    alias, err.response['Error']['Message'])
            else:
                print(f"Deleted alias {alias}.")
        else:
            print("Skipping alias deletion.")
```

- For API details, see [DeleteAlias](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a key

The following code example shows how to describe a KMS key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyManager:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
        self.created_keys = []  
  
    def describe_key(self):  
        """  
        Describes a key.  
        """  
        key_id = input("Enter a key ID or ARN here to get information about the key: ")  
        if key_id:  
            try:  
                key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']  
            except ClientError as err:  
                logging.error(  
                    "Couldn't get key '%s'. Here's why: %s",  
                    key_id, err.response['Error']['Message'])  
            else:  
                print(f"Got key {key_id}:")  
                pprint(key)  
        return key_id
```

- For API details, see [DescribeKey](#) in *AWS SDK for Python (Boto3) API Reference*.

## Disable a key

The following code example shows how to disable a KMS key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyManager:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
        self.created_keys = []  
  
    def enable_disable_key(self, key_id):  
        """  
        Disables and then enables a key. Gets the key state after each state change.  
        """  
        answer = input("Do you want to disable and then enable that key (y/n)? ")  
        if answer.lower() == 'y':  
            try:                # Disable the key  
                self.kms_client.disable_key(KeyId=key_id)  
                # Enable the key  
                self.kms_client.enable_key(KeyId=key_id)
```

```
        self.kms_client.disable_key(KeyId=key_id)
        key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']
    except ClientError as err:
        logging.error(
            "Couldn't disable key '%s'. Here's why: %s",
            key_id, err.response['Error']['Message'])
    else:
        print(f"AWS KMS says your key state is: {key['KeyState']}.")

    try:
        self.kms_client.enable_key(KeyId=key_id)
        key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']
    except ClientError as err:
        logging.error(
            "Couldn't enable key '%s'. Here's why: %s",
            key_id, err.response['Error']['Message'])
    else:
        print(f"AWS KMS says your key state is: {key['KeyState']}.)
```

- For API details, see [DisableKey](#) in *AWS SDK for Python (Boto3) API Reference*.

## Enable a key

The following code example shows how to enable a KMS key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    def enable_disable_key(self, key_id):
        """
        Disables and then enables a key. Gets the key state after each state change.
        """
        answer = input("Do you want to disable and then enable that key (y/n)? ")
        if answer.lower() == 'y':
            try:
                self.kms_client.disable_key(KeyId=key_id)
                key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']
            except ClientError as err:
                logging.error(
                    "Couldn't disable key '%s'. Here's why: %s",
                    key_id, err.response['Error']['Message'])
            else:
                print(f"AWS KMS says your key state is: {key['KeyState']}.")

            try:
                self.kms_client.enable_key(KeyId=key_id)
                key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']
            except ClientError as err:
                logging.error(
                    "Couldn't enable key '%s'. Here's why: %s",
                    key_id, err.response['Error']['Message'])
            else:
                print(f"AWS KMS says your key state is: {key['KeyState']}.)
```

- For API details, see [EnableKey](#) in *AWS SDK for Python (Boto3) API Reference*.

## Encrypt text using a key

The following code example shows how to encrypt text using a KMS key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyEncrypt:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
  
    def encrypt(self, key_id):  
        """  
        Encrypts text by using the specified key.  
  
        :param key_id: The ARN or ID of the key to use for encryption.  
        :return: The encrypted version of the text.  
        """  
        text = input("Enter some text to encrypt: ")  
        try:  
            cipher_text = self.kms_client.encrypt(  
               KeyId=key_id, Plaintext=text.encode())['CiphertextBlob']  
        except ClientError as err:  
            logger.error(  
                "Couldn't encrypt text. Here's why: %s", err.response['Error']  
                ['Message'])  
        else:  
            print(f"Your ciphertext is: {cipher_text}")  
            return cipher_text
```

- For API details, see [Encrypt](#) in *AWS SDK for Python (Boto3) API Reference*.

## Generate a plaintext data key for client-side encryption

The following code example shows how to generate a unique symmetric data key for client-side encryption from an AWS KMS key. The key contains both plaintext and ciphertext versions.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyManager:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
        self.created_keys = []  
  
    def generate_data_key(self, key_id):  
        """
```

```
Generates a symmetric data key that can be used for client-side encryption.  
"""  
answer = input(  
    f"Do you want to generate a symmetric data key from key {key_id} (y/n)? ")  
if answer.lower() == 'y':  
    try:  
        data_key = self.kms_client.generate_data_key(KeyId=key_id,  
KeySpec='AES_256')  
    except ClientError as err:  
        logger.error(  
            "Couldn't generate a data key for key %. Here's why: %s",  
            key_id, err.response['Error']['Message'])  
    else:  
        pprint(data_key)
```

- For API details, see [GenerateDataKey](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a policy for a key

The following code example shows how to get a policy by name for a KMS key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyPolicy:  
    def __init__(self, kms_client):  
        self.kms_client = kms_client  
  
    def get_policy(self, key_id):  
        """  
        Gets the policy of a key.  
  
        :param key_id: The ARN or ID of the key to query.  
        :return: The key policy as a dict.  
        """  
        if key_id != '':  
            try:  
                response = self.kms_client.get_key_policy(  
                    KeyId=key_id, PolicyName='default')  
                policy = json.loads(response['Policy'])  
            except ClientError as err:  
                logger.error(  
                    "Couldn't get policy for key %. Here's why: %s",  
                    key_id, err.response['Error']['Message'])  
            else:  
                pprint(policy)  
                return policy  
        else:  
            print("Skipping get policy demo.")
```

- For API details, see [GetKeyPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## List aliases for a key

The following code example shows how to list aliases for a KMS key.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AliasManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_key = None

    def list_aliases(self):
        """
        Lists aliases for the current account.
        """
        answer = input("\nLet's list your key aliases. Ready (y/n)? ")
        if answer.lower() == 'y':
            try:
                page_size = 10

                aliasPaginator = self.kms_client.getPaginator('list_aliases')
                for alias_page in aliasPaginator.paginate(PaginationConfig={'PageSize': 10}):
                    print(f"Here are {page_size} aliases:")
                    pprint(alias_page['Aliases'])
                    if alias_page['Truncated']:
                        answer = input(
                            f"Do you want to see the next {page_size} aliases (y/n)? ")
                        if answer.lower() != 'y':
                            break
                    else:
                        print("That's all your aliases!")
            except ClientError as err:
                logging.error(
                    "Couldn't list your aliases. Here's why: %s",
                    err.response['Error']['Message'])
```

- For API details, see [ListAliases](#) in *AWS SDK for Python (Boto3) API Reference*.

## List grants for a key

The following code example shows how to list grants for a KMS key.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    def list_grants(self, key_id):
        """
        Lists grants for a key.

        :param key_id: The ARN or ID of the key to query.
        :return: The grants for the key.
        """

```

```
answer = input(f"Ready to list grants on key {key_id} (y/n)? ")
if answer.lower() == 'y':
    try:
        grants = self.kms_client.list_grants(KeyId=key_id)['Grants']
    except ClientError as err:
        logger.error(
            "Couldn't list grants for key %s. Here's why: %s",
            key_id, err.response['Error']['Message'])
    else:
        print(f"Grants for key {key_id}:")
        pprint(grants)
        return grants
```

- For API details, see [ListGrants](#) in *AWS SDK for Python (Boto3) API Reference*.

## List keys

The following code example shows how to list KMS keys.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    def list_keys(self):
        """
        Lists the keys for the current account by using a paginator.
        """
        try:
            page_size = 10
            print("\nLet's list your keys.")
            keyPaginator = self.kms_client.get_paginator('list_keys')
            for key_page in keyPaginator.paginate(PaginationConfig={'PageSize': 10}):
                print(f"Here are {len(key_page['Keys'])} keys:")
                pprint(key_page['Keys'])
                if key_page['Truncated']:
                    answer = input(f"Do you want to see the next {page_size} keys (y/n)? ")
                    if answer.lower() != 'y':
                        break
                else:
                    print("That's all your keys!")
        except ClientError as err:
            logging.error(
                "Couldn't list your keys. Here's why: %s", err.response['Error']['Message'])
```

- For API details, see [ListKeys](#) in *AWS SDK for Python (Boto3) API Reference*.

## List policies for a key

The following code example shows how to list policies for a KMS key.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    def list_policies(self, key_id):
        """
        Lists the names of the policies for a key.

        :param key_id: The ARN or ID of the key to query.
        """
        try:
            policy_names = self.kms_client.list_key_policies(KeyId=key_id)
        except ClientError as err:
            logging.error(
                "Couldn't list your policies. Here's why: %s", err.response['Error'])
        else:
            print(f"The policies for key {key_id} are:")
            pprint(policy_names)
```

- For API details, see [ListKeyPolicies](#) in *AWS SDK for Python (Boto3) API Reference*.

## Recencrypt ciphertext from one key to another

The following code example shows how to reencrypt ciphertext from one KMS key to another.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    def re_encrypt(self, source_key_id, cipher_text):
        """
        Takes ciphertext previously encrypted with one key and reencrypt it by using
        another key.

        :param source_key_id: The ARN or ID of the original key used to encrypt the
                             ciphertext.
        :param cipher_text: The encrypted ciphertext.
        :return: The ciphertext encrypted by the second key.
        """

        destination_key_id = input(
            f"Your ciphertext is currently encrypted with key {source_key_id}. "
            f"Enter another key ID or ARN to reencrypt it: ")
        if destination_key_id != '':
            try:
                cipher_text = self.kms_client.re_encrypt(
                    SourceKeyId=source_key_id, DestinationKeyId=destination_key_id,
```

```
CiphertextBlob=cipher_text)['CiphertextBlob']
except ClientError as err:
    logger.error(
        "Couldn't reencrypt your ciphertext. Here's why: %s",
        err.response['Error']['Message'])
else:
    print(f'Reencrypted your ciphertext as: {cipher_text}')
return cipher_text
else:
    print("Skipping reencryption demo.")
```

- For API details, see [ReEncrypt](#) in *AWS SDK for Python (Boto3) API Reference*.

## Retire a grant for a key

The following code example shows how to retire a grant for a KMS key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    def retire_grant(self, grant):
        """
        Retires a grant so that it can no longer be used.

        :param grant: The grant to retire.
        """
        try:
            self.kms_client.retire_grant(GrantToken=grant['GrantToken'])
        except ClientError as err:
            logger.error(
                "Couldn't retire grant %s. Here's why: %s",
                grant['GrantId'], err.response['Error']['Message'])
        else:
            print(f"Grant {grant['GrantId']} retired.")
```

- For API details, see [RetireGrant](#) in *AWS SDK for Python (Boto3) API Reference*.

## Revoke a grant for a key

The following code example shows how to revoke a grant for a KMS key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GrantManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
```

```
def revoke_grant(self, key_id, grant):
    """
    Revokes a grant so that it can no longer be used.

    :param key_id: The ARN or ID of the key associated with the grant.
    :param grant: The grant to revoke.
    """
    try:
        self.kms_client.revoke_grant(KeyId=key_id, GrantId=grant['GrantId'])
    except ClientError as err:
        logger.error(
            "Couldn't revoke grant %s. Here's why: %s",
            grant['GrantId'], err.response['Error']['Message'])
    else:
        print(f"Grant {grant['GrantId']} revoked.")
```

- For API details, see [RevokeGrant in AWS SDK for Python \(Boto3\) API Reference](#).

## Schedule deletion of a key

The following code example shows how to schedule deletion of a KMS key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    def delete_keys(self, keys):
        """
        Deletes a list of keys.

        :param keys: The list of keys to delete.
        """
        answer = input("Do you want to delete these keys (y/n)? ")
        if answer.lower() == 'y':
            window = 7
            for key in keys:
                try:
                    self.kms_client.schedule_key_deletion(
                        KeyId=key['KeyId'], PendingWindowInDays=window)
                except ClientError as err:
                    logging.error(
                        "Couldn't delete key %s. Here's why: %s",
                        key['KeyId'], err.response['Error']['Message'])
                else:
                    print(f"Key {key['KeyId']} scheduled for deletion in {window} days.")
```

- For API details, see [ScheduleKeyDeletion in AWS SDK for Python \(Boto3\) API Reference](#).

## Set the policy for a key

The following code example shows how to set the policy for a KMS key.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyPolicy:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    def set_policy(self, key_id, policy):
        """
        Sets the policy of a key. Setting a policy entirely overwrites the existing
        policy, so care is taken to add a statement to the existing list of statements
        rather than simply writing a new policy.

        :param key_id: The ARN or ID of the key to set the policy to.
        :param policy: The existing policy of the key.
        """
        user = input("Enter the ARN of an IAM user to set as the principal on the
policy: ")
        if key_id != '' and user != '':
            # The updated policy replaces the existing policy. Add a new statement to
            # the list along with the original policy statements.
            policy['Statement'].append({
                "Sid": "Allow access for ExampleUser",
                "Effect": "Allow",
                "Principal": {"AWS": user},
                "Action": [
                    "kms:Encrypt",
                    "kms:GenerateDataKey*",
                    "kms:Decrypt",
                    "kms:DescribeKey",
                    "kms:ReEncrypt*"],
                "Resource": "*"})
        try:
            self.kms_client.put_key_policy(
                KeyId=key_id, PolicyName='default', Policy=json.dumps(policy))
        except ClientError as err:
            logger.error(
                "Couldn't set policy for key %. Here's why %s",
                key_id, err.response['Error']['Message'])
        else:
            print(f"Set policy for key {key_id}.")
        else:
            print("Skipping set policy demo.")
```

- For API details, see [PutKeyPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update the key referred to by an alias

The following code example shows how to update the KMS key referred to by an alias.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class AliasManager:
```

```
def __init__(self, kms_client):
    self.kms_client = kms_client
    self.created_key = None

def update_alias(self, alias, current_key_id):
    """
    Updates an alias by assigning it to another key.

    :param alias: The alias to reassign.
    :param current_key_id: The ARN or ID of the key currently associated with the
    alias.
    """
    new_key_id = input(
        f"Alias {alias} is currently associated with {current_key_id}. "
        f"Enter another key ID or ARN that you want to associate with {alias}: ")
    if new_key_id != '':
        try:
            self.kms_client.update_alias(AliasName=alias, TargetKeyId=new_key_id)
        except ClientError as err:
            logger.error(
                "Couldn't associate alias %s with key %s. Here's why: %s",
                alias, new_key_id, err.response['Error']['Message'])
        else:
            print(f"Alias {alias} is now associated with key {new_key_id}.")
    else:
        print("Skipping alias update.")
```

- For API details, see [UpdateAlias](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Encrypt and decrypt text

The following code example shows how to:

- Encrypt plain text by using a KMS key.
- Decrypt ciphertext by using a KMS key.
- Reencrypt ciphertext by using a second KMS key.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class KeyEncrypt:
    def __init__(self, kms_client):
        self.kms_client = kms_client

    def encrypt(self, key_id):
        """
        Encrypts text by using the specified key.
        
```

```
:param key_id: The ARN or ID of the key to use for encryption.  
:return: The encrypted version of the text.  
"""  
text = input("Enter some text to encrypt: ")  
try:  
    cipher_text = self.kms_client.encrypt(  
        KeyId=key_id, Plaintext=text.encode())['CiphertextBlob']  
except ClientError as err:  
    logger.error(  
        "Couldn't encrypt text. Here's why: %s", err.response['Error'])  
['Message'])  
else:  
    print(f"Your ciphertext is: {cipher_text}")  
    return cipher_text  
  
def decrypt(self, key_id, cipher_text):  
    """  
    Decrypts text previously encrypted with a key.  
  
    :param key_id: The ARN or ID of the key used to decrypt the data.  
    :param cipher_text: The encrypted text to decrypt.  
    """  
    answer = input("Ready to decrypt your ciphertext (y/n)? ")  
    if answer.lower() == 'y':  
        try:  
            text = self.kms_client.decrypt(  
                KeyId=key_id, CiphertextBlob=cipher_text)['Plaintext']  
        except ClientError as err:  
            logger.error(  
                "Couldn't decrypt your ciphertext. Here's why: %s",  
                err.response['Error']['Message'])  
        else:  
            print(f"Your plaintext is {text.decode()}")  
    else:  
        print("Skipping decryption demo.")  
  
def re_encrypt(self, source_key_id, cipher_text):  
    """  
    Takes ciphertext previously encrypted with one key and reencrypt it by using  
    another key.  
  
    :param source_key_id: The ARN or ID of the original key used to encrypt the  
        ciphertext.  
    :param cipher_text: The encrypted ciphertext.  
    :return: The ciphertext encrypted by the second key.  
    """  
    destination_key_id = input(  
        f"Your ciphertext is currently encrypted with key {source_key_id}. "  
        f"Enter another key ID or ARN to reencrypt it: ")  
    if destination_key_id != '':  
        try:  
            cipher_text = self.kms_client.re_encrypt(  
                SourceKeyId=source_key_id, DestinationKeyId=destination_key_id,  
                CiphertextBlob=cipher_text)['CiphertextBlob']  
        except ClientError as err:  
            logger.error(  
                "Couldn't reencrypt your ciphertext. Here's why: %s",  
                err.response['Error']['Message'])  
        else:  
            print(f"Reencrypted your ciphertext as: {cipher_text}")  
            return cipher_text  
    else:  
        print("Skipping reencryption demo.")
```

```
def key_encryption(kms_client):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the AWS Key Management Service (AWS KMS) key encryption demo.")
    print('*'*88)

    key_id = input("Enter a key ID or ARN to start the demo: ")
    if key_id == '':
        print("A key is required to run this demo.")
        return

    key_encrypt = KeyEncrypt(kms_client)
    cipher_text = key_encrypt.encrypt(key_id)
    print('*'*88)
    if cipher_text is not None:
        key_encrypt.decrypt(key_id, cipher_text)
        print('*'*88)
        key_encrypt.re_encrypt(key_id, cipher_text)

    print("\nThanks for watching!")
    print('*'*88)

if __name__ == '__main__':
    try:
        key_encryption(boto3.client('kms'))
    except Exception:
        logging.exception("Something went wrong with the demo!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [Decrypt](#)
  - [Encrypt](#)
  - [ReEncrypt](#)

## Manage keys

The following code example shows how to:

- Create a KMS key.
- List KMS keys for your account and get details about them.
- Enable and disable KMS keys.
- Generate a symmetric data key that can be used for client-side encryption.
- Delete KMS keys.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
class KeyManager:
    def __init__(self, kms_client):
        self.kms_client = kms_client
        self.created_keys = []

    def create_key(self):
        """
        Creates a key (or multiple keys) with a user-provided description.
        """
        answer = 'y'
        while answer.lower() == 'y':
            key_desc = input("\nLet's create a key. Describe it for me: ")
            if not key_desc:
                key_desc = "Key management demo key"
            try:
                key = self.kms_client.create_key(Description=key_desc)['KeyMetadata']
            except ClientError as err:
                logging.error(
                    "Couldn't create your key. Here's why: %s",
                    err.response['Error']['Message'])
                raise
            else:
                print("Key created:")
                pprint(key)
                self.created_keys.append(key)
                answer = input("Create another (y/n)? ")

    def list_keys(self):
        """
        Lists the keys for the current account by using a paginator.
        """
        try:
            page_size = 10
            print("\nLet's list your keys.")
            keyPaginator = self.kms_client.get_paginator('list_keys')
            for key_page in keyPaginator.paginate(PaginationConfig={'PageSize': 10}):
                print(f"Here are {len(key_page['Keys'])} keys:")
                pprint(key_page['Keys'])
                if key_page['Truncated']:
                    answer = input(f"Do you want to see the next {page_size} keys (y/n)? ")
                    if answer.lower() != 'y':
                        break
                else:
                    print("That's all your keys!")
        except ClientError as err:
            logging.error(
                "Couldn't list your keys. Here's why: %s", err.response['Error']['Message'])

    def describe_key(self):
        """
        Describes a key.
        """
        key_id = input("Enter a key ID or ARN here to get information about the key: ")
        if key_id:
            try:
                key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']
            except ClientError as err:
                logging.error(
                    "Couldn't get key '%s'. Here's why: %s",
                    key_id, err.response['Error']['Message'])
            else:
                print(f"Got key {key_id}:")
                pprint(key)
```

```
    return key_id

def generate_data_key(self, key_id):
    """
    Generates a symmetric data key that can be used for client-side encryption.
    """
    answer = input(
        f"Do you want to generate a symmetric data key from key {key_id} (y/n)? ")
    if answer.lower() == 'y':
        try:
            data_key = self.kms_client.generate_data_key(KeyId=key_id,
KeySpec='AES_256')
        except ClientError as err:
            logger.error(
                "Couldn't generate a data key for key %s. Here's why: %s",
                key_id, err.response['Error']['Message'])
    else:
        pprint(data_key)

def enable_disable_key(self, key_id):
    """
    Disables and then enables a key. Gets the key state after each state change.
    """
    answer = input("Do you want to disable and then enable that key (y/n)? ")
    if answer.lower() == 'y':
        try:
            self.kms_client.disable_key(KeyId=key_id)
            key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']
        except ClientError as err:
            logging.error(
                "Couldn't disable key '%s'. Here's why: %s",
                key_id, err.response['Error']['Message'])
        else:
            print(f"AWS KMS says your key state is: {key['KeyState']}.")

        try:
            self.kms_client.enable_key(KeyId=key_id)
            key = self.kms_client.describe_key(KeyId=key_id)['KeyMetadata']
        except ClientError as err:
            logging.error(
                "Couldn't enable key '%s'. Here's why: %s",
                key_id, err.response['Error']['Message'])
        else:
            print(f"AWS KMS says your key state is: {key['KeyState']}.")

def delete_keys(self, keys):
    """
    Deletes a list of keys.

    :param keys: The list of keys to delete.
    """
    answer = input("Do you want to delete these keys (y/n)? ")
    if answer.lower() == 'y':
        window = 7
        for key in keys:
            try:
                self.kms_client.schedule_key_deletion(
                    KeyId=key['KeyId'], PendingWindowInDays=window)
            except ClientError as err:
                logging.error(
                    "Couldn't delete key %s. Here's why: %s",
                    key['KeyId'], err.response['Error']['Message'])
            else:
                print(f"Key {key['KeyId']} scheduled for deletion in {window}
days.")
```

```
def key_management(kms_client):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the AWS Key Management Service (AWS KMS) key management demo.")
    print('*'*88)

    key_manager = KeyManager(kms_client)
    key_manager.create_key()
    print('*'*88)
    key_manager.list_keys()
    print('*'*88)
    key_id = key_manager.describe_key()
    if key_id:
        key_manager.enable_disable_key(key_id)
        print('*'*88)
        key_manager.generate_data_key(key_id)
    print('*'*88)
    print("For this demo, we created these keys:")
    for key in key_manager.created_keys:
        print(f"\tKeyId: {key['KeyId']}")
        print(f"\tDescription: {key['Description']}")
        print('*'*66)
    key_manager.delete_keys(key_manager.created_keys)
    print("\nThanks for watching!")
    print('*'*88)

if __name__ == '__main__':
    try:
        key_management(boto3.client('kms'))
    except Exception:
        logging.exception("Something went wrong with the demo!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateKey](#)
  - [DescribeKey](#)
  - [DisableKey](#)
  - [EnableKey](#)
  - [GenerateDataKey](#)
  - [ListKeys](#)
  - [ScheduleKeyDeletion](#)

## Amazon Keyspaces examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Keyspaces (for Apache Cassandra).

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Get started

#### [Hello Amazon Keyspaces](#)

The following code example shows how to get started using Amazon Keyspaces (for Apache Cassandra).

## Python

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import boto3

def hello_keyspaces(keyspaces_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Keyspaces (for Apache
    Cassandra) client and list the keyspaces in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param keyspaces_client: A Boto3 Amazon Keyspaces Client object. This object
    wraps
                           the low-level Amazon Keyspaces service API.
    """
    print("Hello, Amazon Keyspaces! Let's list some of your keyspaces:\n")
    for ks in keyspaces_client.list_keyspaces(maxResults=5).get('keyspaces', []):
        print(ks['keyspaceName'])
        print(f"\t{ks['resourceArn']}")

if __name__ == '__main__':
    hello_keyspaces(boto3.client('keyspaces'))
```

- For API details, see [ListKeyspaces](#) in *AWS SDK for Python (Boto3) API Reference*.

#### Topics

- [Actions \(p. 4026\)](#)
- [Scenarios \(p. 4034\)](#)

## Actions

### Create a keyspace

The following code example shows how to create an Amazon Keyspaces keyspace.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:
    """
    Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions.
    """
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
```

```
    self.table_name = None

    @classmethod
    def from_client(cls):
        keyspace_client = boto3.client('keyspaces')
        return cls(keyspace_client)

    def create_keyspace(self, name):
        """
        Creates a keyspace.

        :param name: The name to give the keyspace.
        :return: The Amazon Resource Name (ARN) of the new keyspace.
        """
        try:
            response = self.keyspace_client.create_keyspace(keyspaceName=name)
            self.ks_name = name
            self.ks_arn = response['resourceArn']
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
        else:
            return self.ks_arn
```

- For API details, see [CreateKeyspace in AWS SDK for Python \(Boto3\) API Reference](#).

## Create a table

The following code example shows how to create an Amazon Keyspaces table.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""
    def __init__(self, keyspace_client):
        """
        :param keyspace_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspace_client = keyspace_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspace_client = boto3.client('keyspaces')
        return cls(keyspace_client)

    def create_table(self, table_name):
        """
        Creates a table in the keyspace.
        The table is created with a schema for storing movie data
        and has point-in-time recovery enabled.
        
```

```
:param table_name: The name to give the table.
:return: The ARN of the new table.
"""
try:
    response = self.keyspaces_client.create_table(
        keyspaceName=self.ks_name, tableName=table_name,
        schemaDefinition={
            'allColumns': [
                {'name': 'title', 'type': 'text'},
                {'name': 'year', 'type': 'int'},
                {'name': 'release_date', 'type': 'timestamp'},
                {'name': 'plot', 'type': 'text'},
            ],
            'partitionKeys': [{name: 'year'}, {name: 'title'}]
        },
        pointInTimeRecovery={'status': 'ENABLED'})
except ClientError as err:
    logger.error(
        "Couldn't create table %s. Here's why: %s: %s", table_name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response['resourceArn']
```

- For API details, see [CreateTable](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a keyspace

The following code example shows how to delete an Amazon Keyspaces keyspace.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client('keyspaces')
        return cls(keyspaces_client)

    def delete_keyspace(self):
        """
        Deletes the keyspace.
        """
        try:
            self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
            self.ks_name = None
        except ClientError as err:
            logger.error(
```

```
        "Couldn't delete keyspace %s. Here's why: %s: %s", self.ks_name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
```

- For API details, see [DeleteKeyspace in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete a table

The following code example shows how to delete an Amazon Keyspaces table.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspace_client = keyspace_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspace_client = boto3.client('keyspace')
        return cls(keyspace_client)

    def delete_table(self):
        """
        Deletes the table from the keyspace.
        """
        try:
            self.keyspace_client.delete_table(
                keyspaceName=self.ks_name, tableName=self.table_name)
            self.table_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s", self.table_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

- For API details, see [DeleteTable in AWS SDK for Python \(Boto3\) API Reference](#).

## Get data about a keyspace

The following code example shows how to get data about an Amazon Keyspaces keyspace.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
actions."""
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspace_client = keyspace_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspace_client = boto3.client('keyspaces')
        return cls(keyspace_client)

    def exists_keyspace(self, name):
        """
        Checks whether a keyspace exists.

        :param name: The name of the keyspace to look up.
        :return: True when the keyspace exists. Otherwise, False.
        """
        try:
            response = self.keyspace_client.get_keyspace(keyspaceName=name)
            self.ks_name = response['keyspaceName']
            self.ks_arn = response['resourceArn']
            exists = True
        except ClientError as err:
            if err.response['Error']['Code'] == 'ResourceNotFoundException':
                logger.info("Keyspace %s does not exist.", name)
                exists = False
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    name,
                    err.response['Error']['Code'],
                    err.response['Error']['Message'])
                raise
        return exists
```

- For API details, see [GetKeyspace](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get data about a table

The following code example shows how to get data about an Amazon Keyspaces table.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
actions."""
    def __init__(self, keyspace_client):
        """
        :param keyspace_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspace_client = keyspace_client
        self.ks_name = None
```

```
    self.ks_arn = None
    self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client('keyspaces')
        return cls(keyspaces_client)

    def get_table(self, table_name):
        """
        Gets data about a table in the keyspace.

        :param table_name: The name of the table to look up.
        :return: Data about the table.
        """
        try:
            response = self.keyspaces_client.get_table(
                keySpaceName=self.ks_name, tableName=table_name)
            self.table_name = table_name
        except ClientError as err:
            if err.response['Error']['Code'] == 'ResourceNotFoundException':
                logger.info("Table %s does not exist.", table_name)
                self.table_name = None
                response = None
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    table_name,
                    err.response['Error']['Code'], err.response['Error']['Message'])
                raise
        return response
```

- For API details, see [GetTable](#) in *AWS SDK for Python (Boto3) API Reference*.

## List keyspaces

The following code example shows how to list Amazon Keyspaces keyspaces.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client('keyspaces')
        return cls(keyspaces_client)

    def list_keyspaces(self, limit):
```

```
"""
Lists the keyspaces in your account.

:param limit: The maximum number of keyspaces to list.
"""
try:
    ks Paginator = self.keyspaces_client.getPaginator('list_keyspaces')
    for page in ksPaginator.paginate(PaginationConfig={'MaxItems': limit}):
        for ks in page['keyspaces']:
            print(ks['keyspaceName'])
            print(f"\t{ks['resourceArn']}")
except ClientError as err:
    logger.error(
        "Couldn't list keyspaces. Here's why: %s: %s",
        err.response['Error']['Code'], err.response['Error']['Message'])
raise
```

- For API details, see [ListKeyspaces](#) in *AWS SDK for Python (Boto3) API Reference*.

## List tables in a keyspace

The following code example shows how to list Amazon Keyspaces tables in a keyspace.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
actions."""
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client('keyspaces')
        return cls(keyspaces_client)

    def list_tables(self):
        """
        Lists the tables in the keyspace.
        """
        try:
            tablePaginator = self.keyspaces_client.getPaginator('list_tables')
            for page in tablePaginator.paginate(keyspaceName=self.ks_name):
                for table in page['tables']:
                    print(table['tableName'])
                    print(f"\t{table['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list tables in keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
```

```
raise
```

- For API details, see [ListTables](#) in *AWS SDK for Python (Boto3) API Reference*.

## Restore a table to a point in time

The following code example shows how to restore an Amazon Keyspaces table to a point in time.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:  
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table  
actions."""  
    def __init__(self, keyspaces_client):  
        """  
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.  
        """  
        self.keyspace_client = keyspaces_client  
        self.ks_name = None  
        self.ks_arn = None  
        self.table_name = None  
  
    @classmethod  
    def from_client(cls):  
        keyspace_client = boto3.client('keyspaces')  
        return cls(keyspace_client)  
  
    def restore_table(self, restore_timestamp):  
        """  
        Restores the table to a previous point in time. The table is restored  
        to a new table in the same keyspace.  
  
        :param restore_timestamp: The point in time to restore the table. This time  
        must be in UTC format.  
        :return: The name of the restored table.  
        """  
        try:  
            restored_table_name = f"{self.table_name}_restored"  
            self.keyspace_client.restore_table(  
                sourceKeyspaceName=self.ks_name, sourceTableName=self.table_name,  
                targetKeyspaceName=self.ks_name, targetTableName=restored_table_name,  
                restoreTimestamp=restore_timestamp)  
        except ClientError as err:  
            logger.error(  
                "Couldn't restore table %s. Here's why: %s: %s", restore_timestamp,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return restored_table_name
```

- For API details, see [RestoreTable](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update a table

The following code example shows how to update an Amazon Keyspaces table.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client('keyspaces')
        return cls(keyspaces_client)

    def update_table(self):
        """
        Updates the schema of the table.

        This example updates a table of movie data by adding a new column
        that tracks whether the movie has been watched.
        """
        try:
            self.keyspaces_client.update_table(
                keySpaceName=self.ks_name, tableName=self.table_name,
                addColumns=[{'name': 'watched', 'type': 'boolean'}])
        except ClientError as err:
            logger.error(
                "Couldn't update table %s. Here's why: %s: %s",
                self.table_name,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
```

- For API details, see [UpdateTable](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Get started with keyspaces and tables

The following code example shows how to:

- Create a keyspace.
- Create a table in the keyspace. The table is configured with a schema to hold movie data and has point-in-time recovery enabled.
- Connect to the keyspace with a connection secured by TLS and authenticated with Signature V4 (SigV4).
- Query the table by adding, retrieving, and updating movie data.
- Update the table by adding a column to track watched movies.
- Restore the table to a previous point in time.
- Delete the table and keyspace.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
class KeyspaceScenario:  
    """Runs an interactive scenario that shows how to get started using Amazon  
    Keyspaces."""  
    def __init__(self, ks_wrapper):  
        """  
        :param ks_wrapper: An object that wraps Amazon Keyspace actions.  
        """  
        self.ks_wrapper = ks_wrapper  
  
    @demo_func  
    def create_keyspace(self):  
        """  
        1. Creates a keyspace.  
        2. Lists up to 10 keyspaces in your account.  
        """  
        print("Let's create a keyspace.")  
        ks_name = q.ask(  
            "Enter a name for your new keyspace.\nThe name can contain only letters, "  
            "numbers and underscores: ", q.non_empty)  
        if self.ks_wrapper.exists_keyspace(ks_name):  
            print(f"A keyspace named {ks_name} exists.")  
        else:  
            ks_arn = self.ks_wrapper.create_keyspace(ks_name)  
            ks_exists = False  
            while not ks_exists:  
                wait(3)  
                ks_exists = self.ks_wrapper.exists_keyspace(ks_name)  
            print(f"Created a new keyspace.\n\t{ks_arn}.")  
        print("The first 10 keyspaces in your account are:\n")  
        self.ks_wrapper.list_keyspaces(10)  
  
    @demo_func  
    def create_table(self):  
        """  
        1. Creates a table in the keyspace. The table is configured with a schema to  
        hold  
            movie data and has point-in-time recovery enabled.  
        2. Waits for the table to be in an active state.  
        3. Displays schema information for the table.  
        4. Lists tables in the keyspace.  
        """  
        print("Let's create a table for movies in your keyspace.")  
        table_name = q.ask("Enter a name for your table: ", q.non_empty)  
        table = self.ks_wrapper.get_table(table_name)  
        if table is not None:  
            print(f"A table named {table_name} already exists in keyspace "  
                 f"{self.ks_wrapper.ks_name}.")  
        else:  
            table_arn = self.ks_wrapper.create_table(table_name)  
            print(f"Created table {table_name}: \n\t{table_arn}")  
            table = {'status': None}  
            print("Waiting for your table to be ready...")  
            while table['status'] != 'ACTIVE':  
                wait(5)  
                table = self.ks_wrapper.get_table(table_name)  
        print(f"Your table is {table['status']}. Its schema is:")  
        pp(table['schemaDefinition'])
```

```
print("\nThe tables in your keyspace are:\n")
self.ks_wrapper.list_tables()

@demo_func
def ensure_tls_cert(self):
    """
    Ensures you have a TLS certificate available to use to secure the connection
    to the keyspace. This function downloads a default certificate or lets you
    specify your own.
    """
    print("To connect to your keyspace, you must have a TLS certificate.")
    print("Checking for TLS certificate...")
    cert_path = os.path.join(os.path.dirname(__file__),
QueryManager.DEFAULT_CERT_FILE)
    if not os.path.exists(cert_path):
        if q.ask(f"Do you want to download one from {QueryManager.CERT_URL}? (y/n)"),
            q.is_yesno):
            cert = requests.get(QueryManager.CERT_URL).text
            with open(cert_path, 'w') as cert_file:
                cert_file.write(cert)
        else:
            cert_path = q.ask(
                "Enter the full path to the TLS certificate you want to use: ",
q.non_empty)
            print(f"Certificate {cert_path} will be used to secure the connection to your
keyspace.")
    return cert_path

@demo_func
def query_table(self, qm):
    """
    1. Adds movies to the table from a sample movie data file.
    2. Gets a list of movies from the table and lets you select one.
    3. Displays more information about the selected movie.
    """
    qm.add_movies(self.ks_wrapper.table_name, '../../resources/sample_files/
movies.json')
    movies = qm.get_movies(self.ks_wrapper.table_name)
    print(f"Added {len(movies)} movies to the table:")
    sel = q.choose("Pick one to learn more about it: ", [m.title for m in movies])
    movie_choice = qm.get_movie(self.ks_wrapper.table_name, movies[sel].title,
movies[sel].year)
    print(movie_choice.title)
    print(f"\tReleased: {movie_choice.release_date}")
    print(f"\tPlot: {movie_choice.plot}")

@demo_func
def update_and_restore_table(self, qm):
    """
    1. Updates the table by adding a column to track watched movies.
    2. Marks some of the movies as watched.
    3. Gets the list of watched movies from the table.
    4. Restores to a movies_restored table at a previous point in time.
    5. Gets the list of movies from the restored table.
    """
    print("Let's add a column to record which movies you've watched.")
    pre_update_timestamp = datetime.utcnow()
    print(f"Recorded the current UTC time of {pre_update_timestamp} so we can
restore the table later.")
    self.ks_wrapper.update_table()
    print("Waiting for your table to update...")
    table = {'status': 'UPDATING'}
    while table['status'] != 'ACTIVE':
        wait(5)
        table = self.ks_wrapper.get_table(self.ks_wrapper.table_name)
```

```

        print("Column 'watched' added to table.")
        q.ask("Let's mark some of the movies as watched. Press Enter when you're ready.\n")
        movies = qm.get_movies(self.ks_wrapper.table_name)
        for movie in movies[:10]:
            qm.watched_movie(self.ks_wrapper.table_name, movie.title, movie.year)
            print(f"Marked {movie.title} as watched.")
        movies = qm.get_movies(self.ks_wrapper.table_name, watched=True)
        print('*'*88)
        print("The watched movies in our table are:\n")
        for movie in movies:
            print(movie.title)
        print('*'*88)
        if q.ask("\nDo you want to restore the table to the way it was before all of these\n"
                "updates? Keep in mind, this can take up to 20 minutes. (y/n) ",
                q.is_yesno):
            starting_table_name = self.ks_wrapper.table_name
            table_name_restored = self.ks_wrapper.restore_table(pre_update_timestamp)
            table = {'status': 'RESTORING'}
            while table['status'] != 'ACTIVE':
                wait(10)
                table = self.ks_wrapper.get_table(table_name_restored)
            print(f"Restored {starting_table_name} to {table_name_restored} "
                  f"at a point in time of {pre_update_timestamp}.")
            movies = qm.get_movies(table_name_restored)
            print("Now the movies in our table are:")
            for movie in movies:
                print(movie.title)

    def cleanup(self):
        """
        1. Deletes the table and waits for it to be removed.
        2. Deletes the keyspace.
        """
        if q.ask(f"Do you want to delete your {self.ks_wrapper.table_name} table and "
                f"{self.ks_wrapper.ks_name} keyspace? (y/n) ", q.is_yesno):
            table_name = self.ks_wrapper.table_name
            self.ks_wrapper.delete_table()
            table = self.ks_wrapper.get_table(table_name)
            print("Waiting for the table to be deleted.")
            while table is not None:
                wait(5)
                table = self.ks_wrapper.get_table(table_name)
            print("Table deleted.")
            self.ks_wrapper.delete_keyspace()
            print("Keyspace deleted. If you chose to restore your table during the "
                  "demo, the original table is also deleted.")

    def run_scenario(self):
        logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

        print('*'*88)
        print("Welcome to the Amazon Keyspaces (for Apache Cassandra) demo.")
        print('*'*88)

        self.create_keyspace()
        self.create_table()
        cert_file_path = self.ensure_tls_cert()
        # Use a context manager to ensure the connection to the keyspace is closed.
        with QueryManager(
                cert_file_path, boto3.DEFAULT_SESSION, self.ks_wrapper.ks_name) as qm:
            self.query_table(qm)
            self.update_and_restore_table(qm)
        self.cleanup()

```

```
    print("\nThanks for watching!")
    print('-'*88)

if __name__ == '__main__':
    try:
        scenario = KeyspaceScenario(KeyspaceWrapper.from_client())
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Define a class that wraps keyspace and table actions.

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspace_client = keyspace_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspace_client = boto3.client('keyspaces')
        return cls(keyspace_client)

    def create_keyspace(self, name):
        """
        Creates a keyspace.

        :param name: The name to give the keyspace.
        :return: The Amazon Resource Name (ARN) of the new keyspace.
        """
        try:
            response = self.keyspace_client.create_keyspace(keyspaceName=name)
            self.ks_name = name
            self.ks_arn = response['resourceArn']
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
        else:
            return self.ks_arn

    def exists_keyspace(self, name):
        """
        Checks whether a keyspace exists.

        :param name: The name of the keyspace to look up.
        :return: True when the keyspace exists. Otherwise, False.
        """
        try:
            response = self.keyspace_client.get_keyspace(keyspaceName=name)
            self.ks_name = response['keyspaceName']
            self.ks_arn = response['resourceArn']
            exists = True
        except ClientError as err:
            if err.response['Error']['Code'] == 'ResourceNotFoundException':
```

```
        logger.info("Keyspace %s does not exist.", name)
        exists = False
    else:
        logger.error(
            "Couldn't verify %s exists. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    return exists

def list_keyspaces(self, limit):
    """
    Lists the keyspaces in your account.

    :param limit: The maximum number of keyspaces to list.
    """
    try:
        ksPaginator = self.keyspaces_client.getPaginator('list_keyspaces')
        for page in ksPaginator.paginate(PaginationConfig={'MaxItems': limit}):
            for ks in page['keyspaces']:
                print(ks['keyspaceName'])
                print(f"\t{ks['resourceArn']}")
    except ClientError as err:
        logger.error(
            "Couldn't list keyspaces. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def create_table(self, table_name):
    """
    Creates a table in the keyspace.
    The table is created with a schema for storing movie data
    and has point-in-time recovery enabled.

    :param table_name: The name to give the table.
    :return: The ARN of the new table.
    """
    try:
        response = self.keyspaces_client.create_table(
            keySpaceName=self.ks_name, tableName=table_name,
            schemaDefinition={
                'allColumns': [
                    {'name': 'title', 'type': 'text'},
                    {'name': 'year', 'type': 'int'},
                    {'name': 'release_date', 'type': 'timestamp'},
                    {'name': 'plot', 'type': 'text'},
                ],
                'partitionKeys': [{'name': 'year'}, {'name': 'title'}]
            },
            pointInTimeRecovery={'status': 'ENABLED'})
    except ClientError as err:
        logger.error(
            "Couldn't create table %s. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response['resourceArn']

def get_table(self, table_name):
    """
    Gets data about a table in the keyspace.

    :param table_name: The name of the table to look up.
    :return: Data about the table.
    """
    try:
        response = self.keyspaces_client.get_table(
```

```
        keyspaceName=self.ks_name, tableName=table_name)
        self.table_name = table_name
    except ClientError as err:
        if err.response['Error']['Code'] == 'ResourceNotFoundException':
            logger.info("Table %s does not exist.", table_name)
            self.table_name = None
            response = None
        else:
            logger.error(
                "Couldn't verify %s exists. Here's why: %s: %s",
                table_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
    return response

def list_tables(self):
    """
    Lists the tables in the keyspace.
    """
    try:
        tablePaginator = self.keyspaces_client.getPaginator('list_tables')
        for page in tablePaginator.paginate(keyspaceName=self.ks_name):
            for table in page['tables']:
                print(table['tableName'])
                print(f"\t{table['resourceArn']}")
    except ClientError as err:
        logger.error(
            "Couldn't list tables in keyspace %s. Here's why: %s: %s",
            self.ks_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def update_table(self):
    """
    Updates the schema of the table.

    This example updates a table of movie data by adding a new column
    that tracks whether the movie has been watched.
    """
    try:
        self.keyspaces_client.update_table(
            keyspaceName=self.ks_name, tableName=self.table_name,
            addColumns=[{'name': 'watched', 'type': 'boolean'}])
    except ClientError as err:
        logger.error(
            "Couldn't update table %s. Here's why: %s: %s",
            self.table_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def restore_table(self, restore_timestamp):
    """
    Restores the table to a previous point in time. The table is restored
    to a new table in the same keyspace.

    :param restore_timestamp: The point in time to restore the table. This time
                             must be in UTC format.
    :return: The name of the restored table.
    """
    try:
        restored_table_name = f"{self.table_name}_restored"
        self.keyspaces_client.restore_table(
            sourceKeyspaceName=self.ks_name, sourceTableName=self.table_name,
            targetKeyspaceName=self.ks_name, targetTableName=restored_table_name,
            restoreTimestamp=restore_timestamp)
    except ClientError as err:
        logger.error(
            "Couldn't restore table %s. Here's why: %s: %s",
            self.table_name,
            err.response['Error']['Code'], err.response['Error']['Message'])


```

```

        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return restored_table_name

def delete_table(self):
    """
    Deletes the table from the keyspace.
    """
    try:
        self.keyspaces_client.delete_table(
            keyspaceName=self.ks_name, tableName=self.table_name)
        self.table_name = None
    except ClientError as err:
        logger.error(
            "Couldn't delete table %s. Here's why: %s: %s",
            self.table_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def delete_keyspace(self):
    """
    Deletes the keyspace.
    """
    try:
        self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
        self.ks_name = None
    except ClientError as err:
        logger.error(
            "Couldn't delete keyspace %s. Here's why: %s: %s",
            self.ks_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

```

Define a class that creates a TLS connection to a keyspace, authenticates with SigV4, and sends CQL queries to a table in the keyspace.

```

class QueryManager:
    """
    Manages queries to an Amazon Keyspaces (for Apache Cassandra) keyspace.
    Queries are secured by TLS and authenticated by using the Signature V4 (SigV4)
    AWS signing protocol. This is more secure than sending username and password
    with a plain-text authentication provider.

    This example downloads a default certificate to secure TLS, or lets you specify
    your own.

    This example uses a table of movie data to demonstrate basic queries.
    """

    DEFAULT_CERT_FILE = 'sf-class2-root.crt'
    CERT_URL = f'https://certs.secureserver.net/repository/sf-class2-root.crt'

    def __init__(self, cert_file_path, boto_session, keyspace_name):
        """
        :param cert_file_path: The path and file name of the certificate used for TLS.
        :param boto_session: A Boto3 session. This is used to acquire your AWS
        credentials.
        :param keyspace_name: The name of the keyspace to connect.
        """

        self.cert_file_path = cert_file_path
        self.boto_session = boto_session
        self.ks_name = keyspace_name
        self.cluster = None
        self.session = None

```

```

def __enter__(self):
    """
    Creates a session connection to the keyspace that is secured by TLS and
    authenticated by SigV4.
    """
    ssl_context = SSLContext(PROTOCOL_TLSv1_2)
    ssl_context.load_verify_locations(self.cert_file_path)
    ssl_context.verify_mode = CERT_REQUIRED
    auth_provider = SigV4AuthProvider(self.boto_session)
    contact_point = f"cassandra.{self.boto_session.region_name}.amazonaws.com"
    exec_profile = ExecutionProfile(
        consistency_level=ConsistencyLevel.LOCAL_QUORUM,
        load_balancing_policy=DCAwareRoundRobinPolicy())
    self.cluster = Cluster(
        [contact_point], ssl_context=ssl_context, auth_provider=auth_provider,
        port=9142, execution_profiles={EXEC_PROFILE_DEFAULT: exec_profile},
        protocol_version=4)
    self.cluster.__enter__()
    self.session = self.cluster.connect(self.ks_name)
    return self

def __exit__(self, *args):
    """
    Exits the cluster. This shuts down all existing session connections.
    """
    self.cluster.__exit__(*args)

def add_movies(self, table_name, movie_file_path):
    """
    Gets movies from a JSON file and adds them to a table in the keyspace.

    :param table_name: The name of the table.
    :param movie_file_path: The path and file name of a JSON file that contains
    movie data.
    """
    with open(movie_file_path, 'r') as movie_file:
        movies = json.loads(movie_file.read())
    stmt = self.session.prepare(
        f"INSERT INTO {table_name} (year, title, release_date, plot) VALUES
        (?, ?, ?, ?);")
    for movie in movies[:20]:
        self.session.execute(stmt, parameters=[
            movie['year'], movie['title'],
            date.fromisoformat(movie['info']['release_date'].partition('T')[0]),
            movie['info']['plot']])

def get_movies(self, table_name, watched=None):
    """
    Gets the title and year of the full list of movies from the table.

    :param table_name: The name of the movie table.
    :param watched: When specified, the returned list of movies is filtered to
        either movies that have been watched or movies that have not
        been watched. Otherwise, all movies are returned.
    :return: A list of movies in the table.
    """
    if watched is None:
        stmt = SimpleStatement(f"SELECT title, year from {table_name}")
        params = None
    else:
        stmt = SimpleStatement(
            f"SELECT title, year from {table_name} WHERE watched = %s ALLOW
            FILTERING")
        params = [watched]
    return self.session.execute(stmt, parameters=params).all()

```

```
def get_movie(self, table_name, title, year):
    """
    Gets a single movie from the table, by title and year.

    :param table_name: The name of the movie table.
    :param title: The title of the movie.
    :param year: The year of the movie's release.
    :return: The requested movie.
    """
    return self.session.execute(
        SimpleStatement(f"SELECT * from {table_name} WHERE title = %s AND year = %s"),
        parameters=[title, year]).one()

def watched_movie(self, table_name, title, year):
    """
    Updates a movie as having been watched.

    :param table_name: The name of the movie table.
    :param title: The title of the movie.
    :param year: The year of the movie's release.
    """
    self.session.execute(
        SimpleStatement(f"UPDATE {table_name} SET watched=true WHERE title = %s AND year = %s"),
        parameters=[title, year])
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateKeyspace](#)
  - [CreateTable](#)
  - [DeleteKeyspace](#)
  - [DeleteTable](#)
  - [GetKeyspace](#)
  - [GetTable](#)
  - [ListKeyspaces](#)
  - [ListTables](#)
  - [RestoreTable](#)
  - [UpdateTable](#)

## Kinesis examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Kinesis.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4044\)](#)

## Actions

### Create a stream

The following code example shows how to create a Kinesis stream.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisStream:  
    """Encapsulates a Kinesis stream."""  
    def __init__(self, kinesis_client):  
        """  
        :param kinesis_client: A Boto3 Kinesis client.  
        """  
        self.kinesis_client = kinesis_client  
        self.name = None  
        self.details = None  
        self.stream_exists_waiter = kinesis_client.get_waiter('stream_exists')  
  
    def create(self, name, wait_until_exists=True):  
        """  
        Creates a stream.  
  
        :param name: The name of the stream.  
        :param wait_until_exists: When True, waits until the service reports that  
            the stream exists, then queries for its metadata.  
        """  
        try:  
            self.kinesis_client.create_stream(StreamName=name, ShardCount=1)  
            self.name = name  
            logger.info("Created stream %s.", name)  
            if wait_until_exists:  
                logger.info("Waiting until exists.")  
                self.stream_exists_waiter.wait(StreamName=name)  
                self.describe(name)  
        except ClientError:  
            logger.exception("Couldn't create stream %s.", name)  
            raise
```

- For API details, see [CreateStream](#) in *AWS SDK for Python (Boto3) API Reference*.

### Delete a stream

The following code example shows how to delete a Kinesis stream.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisStream:  
    """Encapsulates a Kinesis stream."""  
    def __init__(self, kinesis_client):
```

```
"""
:param kinesis_client: A Boto3 Kinesis client.
"""
self.kinesis_client = kinesis_client
self.name = None
self.details = None
self.stream_exists_waiter = kinesis_client.get_waiter('stream_exists')

def delete(self):
    """
    Deletes a stream.
    """
    try:
        self.kinesis_client.delete_stream(StreamName=self.name)
        self._clear()
        logger.info("Deleted stream %s.", self.name)
    except ClientError:
        logger.exception("Couldn't delete stream %s.", self.name)
        raise
```

- For API details, see [DeleteStream](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a stream

The following code example shows how to describe a Kinesis stream.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisStream:
    """Encapsulates a Kinesis stream."""
    def __init__(self, kinesis_client):
        """
        :param kinesis_client: A Boto3 Kinesis client.
        """
        self.kinesis_client = kinesis_client
        self.name = None
        self.details = None
        self.stream_exists_waiter = kinesis_client.get_waiter('stream_exists')

    def describe(self, name):
        """
        Gets metadata about a stream.

        :param name: The name of the stream.
        :return: Metadata about the stream.
        """
        try:
            response = self.kinesis_client.describe_stream(StreamName=name)
            self.name = name
            self.details = response['StreamDescription']
            logger.info("Got stream %s.", name)
        except ClientError:
            logger.exception("Couldn't get %s.", name)
            raise
        else:
            return self.details
```

- For API details, see [DescribeStream in AWS SDK for Python \(Boto3\) API Reference](#).

## Get data in batches from a stream

The following code example shows how to get data in batches from a Kinesis stream.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisStream:  
    """Encapsulates a Kinesis stream."""  
    def __init__(self, kinesis_client):  
        """  
        :param kinesis_client: A Boto3 Kinesis client.  
        """  
        self.kinesis_client = kinesis_client  
        self.name = None  
        self.details = None  
        self.stream_exists_waiter = kinesis_client.get_waiter('stream_exists')  
  
    def get_records(self, max_records):  
        """  
        Gets records from the stream. This function is a generator that first gets  
        a shard iterator for the stream, then uses the shard iterator to get records  
        in batches from the stream. Each batch of records is yielded back to the  
        caller until the specified maximum number of records has been retrieved.  
  
        :param max_records: The maximum number of records to retrieve.  
        :return: Yields the current batch of retrieved records.  
        """  
        try:  
            response = self.kinesis_client.get_shard_iterator(  
                StreamName=self.name, ShardId=self.details['Shards'][0]['ShardId'],  
                ShardIteratorType='LATEST')  
            shard_iter = response['ShardIterator']  
            record_count = 0  
            while record_count < max_records:  
                response = self.kinesis_client.get_records(  
                    ShardIterator=shard_iter, Limit=10)  
                shard_iter = response['NextShardIterator']  
                records = response['Records']  
                logger.info("Got %s records.", len(records))  
                record_count += len(records)  
                yield records  
        except ClientError:  
            logger.exception("Couldn't get records from stream %s.", self.name)  
            raise
```

- For API details, see the following topics in [AWS SDK for Python \(Boto3\) API Reference](#).
  - [GetRecords](#)
  - [GetShardIterator](#)

## Put data into a stream

The following code example shows how to put data into a Kinesis stream.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisStream:  
    """Encapsulates a Kinesis stream."""  
    def __init__(self, kinesis_client):  
        """  
        :param kinesis_client: A Boto3 Kinesis client.  
        """  
        self.kinesis_client = kinesis_client  
        self.name = None  
        self.details = None  
        self.stream_exists_waiter = kinesis_client.get_waiter('stream_exists')  
  
    def put_record(self, data, partition_key):  
        """  
        Puts data into the stream. The data is formatted as JSON before it is passed  
        to the stream.  
  
        :param data: The data to put in the stream.  
        :param partition_key: The partition key to use for the data.  
        :return: Metadata about the record, including its shard ID and sequence number.  
        """  
        try:  
            response = self.kinesis_client.put_record(  
                StreamName=self.name,  
                Data=json.dumps(data),  
                PartitionKey=partition_key)  
            logger.info("Put record in stream %s.", self.name)  
        except ClientError:  
            logger.exception("Couldn't put record in stream %s.", self.name)  
            raise  
        else:  
            return response
```

- For API details, see [PutRecord](#) in *AWS SDK for Python (Boto3) API Reference*.

## Kinesis Data Analytics examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Kinesis Data Analytics.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4047\)](#)
- [Data generator \(p. 4055\)](#)

## Actions

### Add an input stream to an application

The following code example shows how to add an input stream to a Kinesis Data Analytics application.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""
    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None

    def add_input(self, input_prefix, stream_arn, input_schema):
        """
        Adds an input stream to the application. The input stream data is mapped
        to an in-application stream that can be processed by your code running in
        Kinesis Data Analytics.

        :param input_prefix: The prefix prepended to in-application input stream names.
        :param stream_arn: The ARN of the input stream.
        :param input_schema: A schema that maps the data in the input stream to the
            runtime environment. This can be automatically generated
            by using `discover_input_schema` or you can create it
            yourself.
        :return: Metadata about the newly added input.
        """
        try:
            response = self.analytics_client.add_application_input(
                ApplicationName=self.name,
                CurrentApplicationVersionId=self.version_id,
                Input={
                    'NamePrefix': input_prefix,
                    'KinesisStreamsInput': {'ResourceARN': stream_arn},
                    'InputSchema': input_schema})
            self.version_id = response['ApplicationVersionId']
            logger.info(
                "Add input stream %s to application %s.", stream_arn, self.name)
        except ClientError:
            logger.exception(
                "Couldn't add input stream %s to application %s.", stream_arn,
                self.name)
            raise
        else:
            return response
```

- For API details, see [AddApplicationInput](#) in *AWS SDK for Python (Boto3) API Reference*.

## Add an output stream to an application

The following code example shows how to add an output stream to a Kinesis Data Analytics application.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""
    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None

    def add_output(self, in_app_stream_name, output_arn):
        """
        Adds an output stream to the application. Kinesis Data Analytics maps data
        from the specified in-application stream to the output stream.

        :param in_app_stream_name: The name of the in-application stream to map
            to the output stream.
        :param output_arn: The ARN of the output stream.
        :return: A list of metadata about the output resources currently assigned
            to the application.
        """
        try:
            response = self.analytics_client.add_application_output(
                ApplicationName=self.name,
                CurrentApplicationVersionId=self.version_id,
                Output={
                    'Name': in_app_stream_name,
                    'KinesisStreamsOutput': {'ResourceARN': output_arn},
                    'DestinationSchema': {'RecordFormatType': 'JSON'})
            outputs = response['OutputDescriptions']
            self.version_id = response['ApplicationVersionId']
            logging.info(
                "Added output %s to %s, which now has %s outputs.", output_arn,
                self.name, len(outputs))
        except ClientError:
            logger.exception("Couldn't add output %s to %s.", output_arn, self.name)
            raise
        else:
            return outputs
```

- For API details, see [AddApplicationOutput](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an application

The following code example shows how to create a Kinesis Data Analytics application.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""
    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
```

```
self.analytics_client = analytics_client
self.name = None
self.arn = None
self.version_id = None
self.create_timestamp = None

def create(self, app_name, role_arn, env='SQL-1_0'):
    """
    Creates a Kinesis Data Analytics application.

    :param app_name: The name of the application.
    :param role_arn: The ARN of a role that can be assumed by Kinesis Data
                     Analytics and grants needed permissions.
    :param env: The runtime environment of the application, such as SQL. Code
                uploaded to the application runs in this environment.
    :return: Metadata about the newly created application.
    """
    try:
        response = self.analytics_client.create_application(
            ApplicationName=app_name, RuntimeEnvironment=env,
            ServiceExecutionRole=role_arn)
        details = response['ApplicationDetail']
        self._update_details(details)
        logger.info("Application %s created.", app_name)
    except ClientError:
        logger.exception("Couldn't create application %s.", app_name)
        raise
    else:
        return details
```

- For API details, see [CreateApplication](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an application

The following code example shows how to delete a Kinesis Data Analytics application.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""
    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None

    def delete(self):
        """
        Deletes an application.
        """
        try:
            self.analytics_client.delete_application(
                ApplicationName=self.name, CreateTimestamp=self.create_timestamp)
            logger.info("Deleted application %s.", self.name)
        except ClientError:
            logger.exception("Couldn't delete application %s.", self.name)
```

```
        except ClientError:
            logger.exception("Couldn't delete application %s.", self.name)
            raise
```

- For API details, see [DeleteApplication](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe an application

The following code example shows how to describe a Kinesis Data Analytics application.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""
    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None

    def describe(self, name):
        """
        Gets metadata about an application.

        :param name: The name of the application to look up.
        :return: Metadata about the application.
        """
        try:
            response = self.analytics_client.describe_application(
                ApplicationName=name)
            details = response['ApplicationDetail']
            self._update_details(details)
            logger.info("Got metadata for application %s.", name)
        except ClientError:
            logger.exception("Couldn't get metadata for application %s.", name)
            raise
        else:
            return details
```

- For API details, see [DescribeApplication](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe an application snapshot

The following code example shows how to describe a Kinesis Data Analytics application snapshot.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""
    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None

    def describe_snapshot(self, application_name, snapshot_name):
        """
        Gets metadata about a previously saved application snapshot.

        :param application_name: The name of the application.
        :param snapshot_name: The name of the snapshot.
        :return: Metadata about the snapshot.
        """
        try:
            response = self.analytics_client.describe_application_snapshot(
                ApplicationName=application_name, SnapshotName=snapshot_name)
            snapshot = response['SnapshotDetails']
            logger.info(
                "Got metadata for snapshot %s of application %s.", snapshot_name,
                application_name)
        except ClientError:
            logger.exception(
                "Couldn't get metadata for snapshot %s of application %s.",
                snapshot_name, application_name)
            raise
        else:
            return snapshot
```

- For API details, see [DescribeApplicationSnapshot](#) in *AWS SDK for Python (Boto3) API Reference*.

## Discover a data format for a stream

The following code example shows how to discover a data format for a Kinesis Data Analytics stream.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:
    """Encapsulates Kinesis Data Analytics application functions."""
    def __init__(self, analytics_client):
        """
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
        """
        self.analytics_client = analytics_client
        self.name = None
        self.arn = None
        self.version_id = None
        self.create_timestamp = None

    def discover_input_schema(self, stream_arn, role_arn):
```

Discovers a schema that maps data in a stream to a format that is usable by an application's runtime environment. The stream must be active and have enough data moving through it for the service to sample. The returned schema can be used when you add the stream as an input to the application or you can write your own schema.

```
:param stream_arn: The ARN of the stream to map.  
:param role_arn: A role that lets Kinesis Data Analytics read from the stream.  
:return: The discovered schema of the data in the input stream.  
"""  
try:  
    response = self.analytics_client.discover_input_schema(  
        ResourceARN=stream_arn,  
        ServiceExecutionRole=role_arn,  
        InputStartingPositionConfiguration={'InputStartingPosition': 'NOW'})  
    schema = response['InputSchema']  
    logger.info("Discovered input schema for stream %s.", stream_arn)  
except ClientError:  
    logger.exception(  
        "Couldn't discover input schema for stream %s.", stream_arn)  
    raise  
else:  
    return schema
```

- For API details, see [DiscoverInputSchema](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start an application

The following code example shows how to start a Kinesis Data Analytics application.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:  
    """Encapsulates Kinesis Data Analytics application functions."""  
    def __init__(self, analytics_client):  
        """  
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.  
        """  
        self.analytics_client = analytics_client  
        self.name = None  
        self.arn = None  
        self.version_id = None  
        self.create_timestamp = None  
  
    def start(self, input_id):  
        """  
        Starts an application. After the application is running, it reads from the  
        specified input stream and runs the application code on the incoming data.  
        :param input_id: The ID of the input to read.  
        """  
        try:  
            self.analytics_client.start_application(  
                ApplicationName=self.name,  
                RunConfiguration={  
                    'SqlRunConfigurations': [{  
                        'InputId': input_id,  
                        'InputStartingPositionConfiguration': {
```

```
        'InputStartingPosition': 'NOW'}]}])  
    logger.info("Started application %s.", self.name)  
except ClientError:  
    logger.exception("Couldn't start application %s.", self.name)  
    raise
```

- For API details, see [StartApplication](#) in *AWS SDK for Python (Boto3) API Reference*.

## Stop an application

The following code example shows how to stop a Kinesis Data Analytics application.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class KinesisAnalyticsApplicationV2:  
    """Encapsulates Kinesis Data Analytics application functions."""  
    def __init__(self, analytics_client):  
        """  
        :param analytics_client: A Boto3 Kinesis Data Analytics v2 client.  
        """  
        self.analytics_client = analytics_client  
        self.name = None  
        self.arn = None  
        self.version_id = None  
        self.create_timestamp = None  
  
    def stop(self):  
        """  
        Stops an application. This stops the application from processing data but  
        does not delete any resources.  
        """  
        try:  
            self.analytics_client.stop_application(ApplicationName=self.name)  
            logger.info("Stopping application %s.", self.name)  
        except ClientError:  
            logger.exception("Couldn't stop application %s.", self.name)  
            raise
```

- For API details, see [StopApplication](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update an application

The following code example shows how to update a Kinesis Data Analytics application.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

This example updates the code that runs in an existing application.

```
class KinesisAnalyticsApplicationV2:  
    """Encapsulates Kinesis Data Analytics application functions."""  
    def __init__(self, analytics_client):
```

```
"""
:param analytics_client: A Boto3 Kinesis Data Analytics v2 client.
"""
self.analytics_client = analytics_client
self.name = None
self.arn = None
self.version_id = None
self.create_timestamp = None

def update_code(self, code):
    """
    Updates the code that runs in the application. The code must run in the
    runtime environment of the application, such as SQL. Application code
    typically reads data from in-application streams and transforms it in some way.

    :param code: The code to upload. This completely replaces any existing code
                 in the application.
    :return: Metadata about the application.
    """
    try:
        response = self.analytics_client.update_application(
            ApplicationName=self.name,
            CurrentApplicationVersionId=self.version_id,
            ApplicationConfigurationUpdate={
                'ApplicationCodeConfigurationUpdate': {
                    'CodeContentTypeUpdate': 'PLAINTEXT',
                    'CodeContentUpdate': [
                        'TextContentUpdate': code]
                }
            })
        details = response['ApplicationDetail']
        self.version_id = details['ApplicationVersionId']
        logger.info("Update code for application %s.", self.name)
    except ClientError:
        logger.exception("Couldn't update code for application %s.", self.name)
        raise
    else:
        return details
```

- For API details, see [UpdateApplication](#) in *AWS SDK for Python (Boto3) API Reference*.

## Data generator

### Generate a stream with a referrer

The following code example shows how to generate a Kinesis stream with a referrer.

For more information, see [Example: Extracting a portion of a stream](#).

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import json
import boto3

STREAM_NAME = 'ExampleInputStream'

def get_data():
    return {'REFERRER': 'http://www.amazon.com'}
```

```
def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey='partitionkey')

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

### Generate a stream with blood pressure anomalies

The following code example shows how to generate a Kinesis stream with blood pressure anomalies.

For more information, see [Example: Detecting data anomalies and getting an explanation](#).

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
from enum import Enum
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

class PressureType(Enum):
    low = 'LOW'
    normal = 'NORMAL'
    high = 'HIGH'

def get_blood_pressure(pressure_type):
    pressure = {'BloodPressureLevel': pressure_type.value}
    if pressure_type == PressureType.low:
        pressure['Systolic'] = random.randint(50, 80)
        pressure['Diastolic'] = random.randint(30, 50)
    elif pressure_type == PressureType.normal:
        pressure['Systolic'] = random.randint(90, 120)
        pressure['Diastolic'] = random.randint(60, 80)
    elif pressure_type == PressureType.high:
        pressure['Systolic'] = random.randint(130, 200)
        pressure['Diastolic'] = random.randint(90, 150)
    else:
        raise TypeError
    return pressure

def generate(stream_name, kinesis_client):
    while True:
        rnd = random.random()
        pressure_type = (
```

```
    PressureType.low if rnd < 0.005
    else PressureType.high if rnd > 0.995
    else PressureType.normal)
blood_pressure = get_blood_pressure(pressure_type)
print(blood_pressure)
kinesis_client.put_record(
    StreamName=stream_name,
    Data=json.dumps(blood_pressure),
    PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

### Generate a stream with data in columns

The following code example shows how to generate a Kinesis stream with data in columns.

For more information, see [Example: Split strings into multiple fields](#).

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'Col_A': 'a',
        'Col_B': 'b',
        'Col_C': 'c',
        'Col_E_Unstructured': 'x,y,z'}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

### Generate a stream with heart rate anomalies

The following code example shows how to generate a Kinesis stream with heart rate anomalies.

For more information, see [Example: Detecting data anomalies on a stream](#).

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
from enum import Enum
import json
import random
import boto3

STREAM_NAME = 'ExampleInputStream'

class RateType(Enum):
    normal = 'NORMAL'
    high = 'HIGH'

def get_heart_rate(rate_type):
    if rate_type == RateType.normal:
        rate = random.randint(60, 100)
    elif rate_type == RateType.high:
        rate = random.randint(150, 200)
    else:
        raise TypeError
    return {'heartRate': rate, 'rateType': rate_type.value}

def generate(stream_name, kinesis_client, output=True):
    while True:
        rnd = random.random()
        rate_type = RateType.high if rnd < 0.01 else RateType.normal
        heart_rate = get_heart_rate(rate_type)
        if output:
            print(heart_rate)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(heart_rate),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

## Generate a stream with hotspots

The following code example shows how to generate a Kinesis stream with hotspots.

For more information, see [Example: Detecting hotspot on a stream](#).

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

import json
from pprint import pprint
import random
import time
import boto3

STREAM_NAME = "ExampleInputStream"

def get_hotspot(field, spot_size):
    hotspot = {
        'left': field['left'] + random.random() * (field['width'] - spot_size),
        'width': spot_size,
        'top': field['top'] + random.random() * (field['height'] - spot_size),
        'height': spot_size
    }
    return hotspot

def get_record(field, hotspot, hotspot_weight):
    rectangle = hotspot if random.random() < hotspot_weight else field
    point = {
        'x': rectangle['left'] + random.random() * rectangle['width'],
        'y': rectangle['top'] + random.random() * rectangle['height'],
        'is_hot': 'Y' if rectangle is hotspot else 'N'
    }
    return {'Data': json.dumps(point), 'PartitionKey': 'partition_key'}

def generate(
        stream_name, field, hotspot_size, hotspot_weight, batch_size, kinesis_client):
    """
    Generates points used as input to a hotspot detection algorithm.
    With probability hotspot_weight (20%), a point is drawn from the hotspot;
    otherwise, it is drawn from the base field. The location of the hotspot
    changes for every 1000 points generated.
    """
    points_generated = 0
    hotspot = None
    while True:
        if points_generated % 1000 == 0:
            hotspot = get_hotspot(field, hotspot_size)
        records = [
            get_record(field, hotspot, hotspot_weight) for _ in range(batch_size)]
        points_generated += len(records)
        pprint(records)
        kinesis_client.put_records(StreamName=stream_name, Records=records)

        time.sleep(0.1)

if __name__ == "__main__":
    generate(
        stream_name=STREAM_NAME,
        field={'left': 0, 'width': 10, 'top': 0, 'height': 10},
        hotspot_size=1, hotspot_weight=0.2, batch_size=10,
        kinesis_client=boto3.client('kinesis'))

```

## Generate a stream with log entries

The following code example shows how to generate a Kinesis stream with log entries.

For more information, see [Example: Parsing log strings based on regular expressions](#).

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import json
import boto3

STREAM_NAME = 'ExampleInputStream'

def get_data():
    return {
        'LOGENTRY': '203.0.113.24 - - [25/Mar/2018:15:25:37 -0700] '
                    '"GET /index.php HTTP/1.1" 200 125 "-" '
                    '"Mozilla/5.0 [en] Gecko/20100101 Firefox/52.0"'
    }

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

## Generate a stream with stagger data

The following code example shows how to generate a Kinesis stream with stagger data.

For more information, see [Example: Stagger window](#).

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import datetime
import json
import random
import time
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    event_time = datetime.datetime.utcnow() - datetime.timedelta(seconds=10)
    return {
        'EVENT_TIME': event_time.isoformat(),
```

```
'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV'])}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        # Send six records, ten seconds apart, with the same event time and ticker
        for _ in range(6):
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey")
            time.sleep(10)

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

### Generate a stream with stock ticker data

The following code example shows how to generate a Kinesis stream with stock ticker data.

For more information, see [Examples: Windows and aggregation](#).

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'EVENT_TIME': datetime.datetime.now().isoformat(),
        'TICKER': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'PRICE': round(random.random() * 100, 2)}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

## Generate a stream with two data types

The following code example shows how to generate a Kinesis stream with two data types.

For more information, see [Example: Transforming multiple data types](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import json
import random
import boto3

STREAM_NAME = "OrdersAndTradesStream"
PARTITION_KEY = "partition_key"

def get_order(order_id, ticker):
    return {
        'RecordType': 'Order',
        'Oid': order_id,
        'Oticker': ticker,
        'Oprice': random.randint(500, 10000),
        'Otype': 'Sell'}

def get_trade(order_id, trade_id, ticker):
    return {
        'RecordType': "Trade",
        'Tid': trade_id,
        'Toid': order_id,
        'Tticker': ticker,
        'Tprice': random.randint(0, 3000)}

def generate(stream_name, kinesis_client):
    order_id = 1
    while True:
        ticker = random.choice(['AAAA', 'BBBB', 'CCCC'])
        order = get_order(order_id, ticker)
        print(order)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(order),
            PartitionKey=PARTITION_KEY)
        for trade_id in range(1, random.randint(0, 6)):
            trade = get_trade(order_id, trade_id, ticker)
            print(trade)
            kinesis_client.put_record(
                StreamName=stream_name, Data=json.dumps(trade),
                PartitionKey=PARTITION_KEY)
        order_id += 1

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

## Generate a stream with web log data

The following code example shows how to generate a Kinesis stream with web log data.

For more information, see [Example: Parsing web logs](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import json
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {'log': '192.168.254.30 - John [24/May/2004:22:01:02 -0700] '
                 '"GET /icons/apache_pb.gif HTTP/1.1" 304 0'}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis'))
```

## Lambda examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with AWS Lambda.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4063\)](#)
- [Scenarios \(p. 4068\)](#)

## Actions

### Create a function

The following code example shows how to create a Lambda function.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    def create_function(self, function_name, handler_name, iam_role,
deployment_package):
        """
        Deploys a Lambda function.

        :param function_name: The name of the Lambda function.
        :param handler_name: The fully qualified name of the handler function. This
            must include the file name and the function name.
        :param iam_role: The IAM role to use for the function.
        :param deployment_package: The deployment package that contains the function
            code in .zip format.
        :return: The Amazon Resource Name (ARN) of the newly created function.
        """
        try:
            response = self.lambda_client.create_function(
                FunctionName=function_name,
                Description="AWS Lambda doc example",
                Runtime='python3.8',
                Role=iam_role.arn,
                Handler=handler_name,
                Code={'ZipFile': deployment_package},
                Publish=True)
            function_arn = response['FunctionArn']
            waiter = self.lambda_client.get_waiter('function_active_v2')
            waiter.wait(FunctionName=function_name)
            logger.info("Created function '%s' with ARN: '%s'.",
                        function_name, response['FunctionArn'])
        except ClientError:
            logger.error("Couldn't create function %s.", function_name)
            raise
        else:
            return function_arn
```

- For API details, see [CreateFunction](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a function

The following code example shows how to delete a Lambda function.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
```

```
    self.iam_resource = iam_resource

def delete_function(self, function_name):
    """
    Deletes a Lambda function.

    :param function_name: The name of the function to delete.
    """
    try:
        self.lambda_client.delete_function(FunctionName=function_name)
    except ClientError:
        logger.exception("Couldn't delete function %s.", function_name)
        raise
```

- For API details, see [DeleteFunction](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a function

The following code example shows how to get a Lambda function.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    def get_function(self, function_name):
        """
        Gets data about a Lambda function.

        :param function_name: The name of the function.
        :return: The function data.
        """
        response = None
        try:
            response = self.lambda_client.get_function(FunctionName=function_name)
        except ClientError as err:
            if err.response['Error']['Code'] == 'ResourceNotFoundException':
                logger.info("Function %s does not exist.", function_name)
            else:
                logger.error(
                    "Couldn't get function %s. Here's why: %s: %s",
                    function_name,
                    err.response['Error']['Code'],
                    err.response['Error']['Message'])
            raise
        return response
```

- For API details, see [GetFunction](#) in *AWS SDK for Python (Boto3) API Reference*.

## Invoke a function

The following code example shows how to invoke a Lambda function.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    def invoke_function(self, function_name, function_params, get_log=False):
        """
        Invokes a Lambda function.

        :param function_name: The name of the function to invoke.
        :param function_params: The parameters of the function as a dict. This dict
                               is serialized to JSON before it is sent to Lambda.
        :param get_log: When true, the last 4 KB of the execution log are included in
                       the response.
        :return: The response from the function invocation.
        """
        try:
            response = self.lambda_client.invoke(
                FunctionName=function_name,
                Payload=json.dumps(function_params),
                LogType='Tail' if get_log else 'None')
            logger.info("Invoked function %s.", function_name)
        except ClientError:
            logger.exception("Couldn't invoke function %s.", function_name)
            raise
        return response
```

- For API details, see [Invoke in AWS SDK for Python \(Boto3\) API Reference](#).

## List functions

The following code example shows how to list Lambda functions.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    def list_functions(self):
        """
        Lists the Lambda functions for the current account.
        """
        try:
            funcPaginator = self.lambda_client.getPaginator('list_functions')
            for funcPage in funcPaginator.paginate():
                for func in funcPage['Functions']:
                    print(func['FunctionName'])
                    desc = func.get('Description')
```

```
        if desc:
            print(f"\t{desc}")
            print(f"\t\t{func['Runtime']}: {func['Handler']}")
    except ClientError as err:
        logger.error(
            "Couldn't list functions. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
```

- For API details, see [ListFunctions](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update function code

The following code example shows how to update Lambda function code.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    def update_function_code(self, function_name, deployment_package):
        """
        Updates the code for a Lambda function by submitting a .zip archive that
        contains
        the code for the function.

        :param function_name: The name of the function to update.
        :param deployment_package: The function code to update, packaged as bytes in
                                  .zip format.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_code(
                FunctionName=function_name, ZipFile=deployment_package)
        except ClientError as err:
            logger.error(
                "Couldn't update function %s. Here's why: %s: %s",
                function_name, err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
        else:
            return response
```

- For API details, see [UpdateFunctionCode](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update function configuration

The following code example shows how to update Lambda function configuration.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    def update_function_configuration(self, function_name, env_vars):
        """
        Updates the environment variables for a Lambda function.

        :param function_name: The name of the function to update.
        :param env_vars: A dict of environment variables to update.
        :return: Data about the update, including the status.
        """
        try:
            response = self.lambda_client.update_function_configuration(
                FunctionName=function_name, Environment={'Variables': env_vars})
        except ClientError as err:
            logger.error(
                "Couldn't update function configuration %s. Here's why: %s: %s",
                function_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Get started with functions

The following code example shows how to:

- Create an AWS Identity and Access Management (IAM) role that grants Lambda permission to write to logs.
- Create a Lambda function and upload handler code.
- Invoke the function with a single parameter and get results.
- Update the function code and configure its Lambda environment with an environment variable.
- Invoke the function with new parameters and get results. Display the execution log that's returned from the invocation.
- List the functions for your account.
- Delete the IAM role and the Lambda function.

For more information, see [Create a Lambda function with the console](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Define a Lambda handler that increments a number.

```
import logging
```

```
logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    """
    Accepts an action and a single number, performs the specified action on the number,
    and returns the result. The only allowable action is 'increment'.

    :param event: The event dict that contains the parameters sent when the function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the action.
    """
    result = None
    action = event.get('action')
    if action == 'increment':
        result = event.get('number', 0) + 1
        logger.info('Calculated result of %s', result)
    else:
        logger.error("%s is not a valid action.", action)

    response = {'result': result}
    return response
```

Define a second Lambda handler that performs arithmetic operations.

```
import logging
import os

logger = logging.getLogger()

# Define a list of Python lambda functions that are called by this AWS Lambda function.
ACTIONS = {
    'plus': lambda x, y: x + y,
    'minus': lambda x, y: x - y,
    'times': lambda x, y: x * y,
    'divided-by': lambda x, y: x / y}

def lambda_handler(event, context):
    """
    Accepts an action and two numbers, performs the specified action on the numbers,
    and returns the result.

    :param event: The event dict that contains the parameters sent when the function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the specified action.
    """
    # Set the log level based on a variable configured in the Lambda environment.
    logger.setLevel(os.environ.get('LOG_LEVEL', logging.INFO))
    logger.debug('Event: %s', event)

    action = event.get('action')
    func = ACTIONS.get(action)
    x = event.get('x')
    y = event.get('y')
    result = None
    try:
        if func is not None and x is not None and y is not None:
            result = func(x, y)
```

```

        logger.info("%s %s %s is %s", x, action, y, result)
    else:
        logger.error("I can't calculate %s %s %s.", x, action, y)
except ZeroDivisionError:
    logger.warning("I can't divide %s by 0!", x)

response = {'result': result}
return response

```

Create functions that wrap Lambda actions.

```

class LambdaWrapper:
    def __init__(self, lambda_client, iam_resource):
        self.lambda_client = lambda_client
        self.iam_resource = iam_resource

    @staticmethod
    def create_deployment_package(source_file, destination_file):
        """
        Creates a Lambda deployment package in .zip format in an in-memory buffer. This
        buffer can be passed directly to Lambda when creating the function.

        :param source_file: The name of the file that contains the Lambda handler
                            function.
        :param destination_file: The name to give the file when it's deployed to
                                Lambda.
        :return: The deployment package.
        """
        buffer = io.BytesIO()
        with zipfile.ZipFile(buffer, 'w') as zipped:
            zipped.write(source_file, destination_file)
        buffer.seek(0)
        return buffer.read()

    def get_iam_role(self, iam_role_name):
        """
        Get an AWS Identity and Access Management (IAM) role.

        :param iam_role_name: The name of the role to retrieve.
        :return: The IAM role.
        """
        role = None
        try:
            temp_role = self.iam_resource.Role(iam_role_name)
            temp_role.load()
            role = temp_role
            logger.info("Got IAM role %s", role.name)
        except ClientError as err:
            if err.response['Error']['Code'] == 'NoSuchEntity':
                logger.info("IAM role %s does not exist.", iam_role_name)
            else:
                logger.error(
                    "Couldn't get IAM role %s. Here's why: %s: %s",
                    iam_role_name,
                    err.response['Error']['Code'],
                    err.response['Error']['Message']
                )
                raise
        return role

    def create_iam_role_for_lambda(self, iam_role_name):
        """
        Creates an IAM role that grants the Lambda function basic permissions. If a
        role with the specified name already exists, it is used for the demo.

        :param iam_role_name: The name of the role to create.
        :return: The role and a value that indicates whether the role is newly created.
        """

```

```
"""
role = self.get_iam_role(iam_role_name)
if role is not None:
    return role, False

lambda_assume_role_policy = {
    'Version': '2012-10-17',
    'Statement': [
        {
            'Effect': 'Allow',
            'Principal': {
                'Service': 'lambda.amazonaws.com'
            },
            'Action': 'sts:AssumeRole'
        }
    ]
}
policy_arn = 'arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole'

try:
    role = self.iam_resource.create_role(
        RoleName=iam_role_name,
        AssumeRolePolicyDocument=json.dumps(lambda_assume_role_policy))
    logger.info("Created role %s.", role.name)
    role.attach_policy(PolicyArn=policy_arn)
    logger.info("Attached basic execution policy to role %s.", role.name)
except ClientError as error:
    if error.response['Error']['Code'] == 'EntityAlreadyExists':
        role = self.iam_resource.Role(iam_role_name)
        logger.warning("The role %s already exists. Using it.", iam_role_name)
    else:
        logger.exception(
            "Couldn't create role %s or attach policy %s.",
            iam_role_name, policy_arn)
        raise

return role, True

def get_function(self, function_name):
    """
    Gets data about a Lambda function.

    :param function_name: The name of the function.
    :return: The function data.
    """
    response = None
    try:
        response = self.lambda_client.get_function(FunctionName=function_name)
    except ClientError as err:
        if err.response['Error']['Code'] == 'ResourceNotFoundException':
            logger.info("Function %s does not exist.", function_name)
        else:
            logger.error(
                "Couldn't get function %s. Here's why: %s: %s",
                function_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
    return response

def create_function(self, function_name, handler_name, iam_role,
deployment_package):
    """
    Deploys a Lambda function.

    :param function_name: The name of the Lambda function.
    :param handler_name: The fully qualified name of the handler function. This
                         must include the file name and the function name.
    """

```

```
:param iam_role: The IAM role to use for the function.
:param deployment_package: The deployment package that contains the function
    code in .zip format.
:return: The Amazon Resource Name (ARN) of the newly created function.
"""
try:
    response = self.lambda_client.create_function(
        FunctionName=function_name,
        Description="AWS Lambda doc example",
        Runtime='python3.8',
        Role=iam_role.arn,
        Handler=handler_name,
        Code={'ZipFile': deployment_package},
        Publish=True)
    function_arn = response['FunctionArn']
    waiter = self.lambda_client.get_waiter('function_active_v2')
    waiter.wait(FunctionName=function_name)
    logger.info("Created function '%s' with ARN: '%s'", function_name, response['FunctionArn'])
except ClientError:
    logger.error("Couldn't create function %s.", function_name)
    raise
else:
    return function_arn

def delete_function(self, function_name):
"""
Deletes a Lambda function.

:param function_name: The name of the function to delete.
"""
try:
    self.lambda_client.delete_function(FunctionName=function_name)
except ClientError:
    logger.exception("Couldn't delete function %s.", function_name)
    raise

def invoke_function(self, function_name, function_params, get_log=False):
"""
Invokes a Lambda function.

:param function_name: The name of the function to invoke.
:param function_params: The parameters of the function as a dict. This dict
    is serialized to JSON before it is sent to Lambda.
:param get_log: When true, the last 4 KB of the execution log are included in
    the response.
:return: The response from the function invocation.
"""
try:
    response = self.lambda_client.invoke(
        FunctionName=function_name,
        Payload=json.dumps(function_params),
        LogType='Tail' if get_log else 'None')
    logger.info("Invoked function %s.", function_name)
except ClientError:
    logger.exception("Couldn't invoke function %s.", function_name)
    raise
return response

def update_function_code(self, function_name, deployment_package):
"""
Updates the code for a Lambda function by submitting a .zip archive that
contains
the code for the function.

:param function_name: The name of the function to update.
```

```

:param deployment_package: The function code to update, packaged as bytes in
                         .zip format.
:return: Data about the update, including the status.
"""
try:
    response = self.lambda_client.update_function_code(
        FunctionName=function_name, ZipFile=deployment_package)
except ClientError as err:
    logger.error(
        "Couldn't update function %s. Here's why: %s: %s",
        function_name,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return response

def update_function_configuration(self, function_name, env_vars):
    """
    Updates the environment variables for a Lambda function.

    :param function_name: The name of the function to update.
    :param env_vars: A dict of environment variables to update.
    :return: Data about the update, including the status.
    """
    try:
        response = self.lambda_client.update_function_configuration(
            FunctionName=function_name, Environment={'Variables': env_vars})
    except ClientError as err:
        logger.error(
            "Couldn't update function configuration %s. Here's why: %s: %s",
            function_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response

def list_functions(self):
    """
    Lists the Lambda functions for the current account.
    """
    try:
        funcPaginator = self.lambda_client.getPaginator('list_functions')
        for funcPage in funcPaginator.paginate():
            for func in funcPage['Functions']:
                print(func['FunctionName'])
                desc = func.get('Description')
                if desc:
                    print(f"\t{desc}")
                    print(f"\t{func['Runtime']}: {func['Handler']}"))
    except ClientError as err:
        logger.error(
            "Couldn't list functions. Here's why: %s: %s",
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

```

Create a function that runs the scenario.

```

class UpdateFunctionWaiter(CustomWaiter):
    """
    A custom waiter that waits until a function is successfully updated.
    """
    def __init__(self, client):
        super().__init__(
            'UpdateSuccess', 'GetFunction',
            'Configuration.LastUpdateStatus',
            {'Successful': WaitState.SUCCESS, 'Failed': WaitState.FAILURE},
            client)

```

```
def wait(self, function_name):
    self._wait(FunctionName=function_name)

def run_scenario(lambda_client, iam_resource, basic_file, calculator_file,
lambda_name):
    """
    Runs the scenario.

    :param lambda_client: A Boto3 Lambda client.
    :param iam_resource: A Boto3 IAM resource.
    :param basic_file: The name of the file that contains the basic Lambda handler.
    :param calculator_file: The name of the file that contains the calculator Lambda
handler.
    :param lambda_name: The name to give resources created for the scenario, such as
the
                           IAM role and the Lambda function.
    """
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the AWS Lambda getting started with functions demo.")
    print('*'*88)

    wrapper = LambdaWrapper(lambda_client, iam_resource)

    print("Checking for IAM role for Lambda...")
    iam_role, should_wait = wrapper.create_iam_role_for_lambda(lambda_name)
    if should_wait:
        logger.info("Giving AWS time to create resources...")
        wait(10)

    print(f"Looking for function {lambda_name}...")
    function = wrapper.get_function(lambda_name)
    if function is None:
        print("Zipping the Python script into a deployment package...")
        deployment_package = wrapper.create_deployment_package(basic_file,
f"{lambda_name}.py")
        print(f"...and creating the {lambda_name} Lambda function.")
        wrapper.create_function(
            lambda_name, f'{lambda_name}.lambda_handler', iam_role, deployment_package)
    else:
        print(f"Function {lambda_name} already exists.")
    print('*'*88)

    print(f"Let's invoke {lambda_name}. This function increments a number.")
    action_params = {
        'action': 'increment',
        'number': q.ask("Give me a number to increment: ", q.is_int)}
    print(f"Invoking {lambda_name}...")
    response = wrapper.invoke_function(lambda_name, action_params)
    print(f"Incrementing {action_params['number']} resulted in "
          f'{json.load(response["Payload"])}')
    print('*'*88)

    print(f"Let's update the function to an arithmetic calculator.")
    q.ask("Press Enter when you're ready.")
    print("Creating a new deployment package...")
    deployment_package = wrapper.create_deployment_package(calculator_file,
f"{lambda_name}.py")
    print(f"...and updating the {lambda_name} Lambda function.")
    update_waiter = UpdateFunctionWaiter(lambda_client)
    wrapper.update_function_code(lambda_name, deployment_package)
    update_waiter.wait(lambda_name)
    print(f"This function uses an environment variable to control logging level.")


```

```

print(f"Let's set it to DEBUG to get the most logging.")
wrapper.update_function_configuration(
    lambda_name, {'LOG_LEVEL': logging.getLoggerName(logging.DEBUG)})

actions = ['plus', 'minus', 'times', 'divided-by']
want_invoke = True
while want_invoke:
    print(f"Let's invoke {lambda_name}. You can invoke these actions:")
    for index, action in enumerate(actions):
        print(f'{index + 1}: {action}')
    action_params = {}
    action_index = q.ask(
        "Enter the number of the action you want to take: ",
        q.is_int, q.in_range(1, len(actions)))
    action_params['action'] = actions[action_index - 1]
    print(f"You've chosen to invoke 'x {action_params['action']} y'.")
    action_params['x'] = q.ask("Enter a value for x: ", q.is_int)
    action_params['y'] = q.ask("Enter a value for y: ", q.is_int)
    print(f"Invoking {lambda_name}...")
    response = wrapper.invoke_function(lambda_name, action_params, True)
    print(f"Calculating {action_params['x']} {action_params['action']} {action_params['y']}")
    f"resulted in {json.loads(response['Payload'])}"
    q.ask("Press Enter to see the logs from the call.")
    print(base64.b64decode(response['LogResult']).decode())
    want_invoke = q.ask("That was fun. Shall we do it again? (y/n)", q.is_yesno)
    print('-'*88)

    if q.ask("Do you want to list all of the functions in your account? (y/n)", q.is_yesno):
        wrapper.list_functions()
        print('-'*88)

    if q.ask("Ready to delete the function and role? (y/n)", q.is_yesno):
        for policy in iam_role.attached_policies.all():
            policy.detach_role(RoleName=iam_role.name)
        iam_role.delete()
        print(f"Deleted role {lambda_name}.")
        wrapper.delete_function(lambda_name)
        print(f"Deleted function {lambda_name}.")

    print("\nThanks for watching!")
    print('-'*88)

if __name__ == '__main__':
    try:
        run_scenario(
            boto3.client('lambda'), boto3.resource('iam'), 'lambda_handler_basic.py',
            'lambda_handler_calculator.py', 'doc_example_lambda_calculator')
    except Exception:
        logging.exception("Something went wrong with the demo!")

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)
  - [Invoke](#)
  - [ListFunctions](#)
  - [UpdateFunctionCode](#)
  - [UpdateFunctionConfiguration](#)

## Lookout for Vision examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Lookout for Vision.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4076\)](#)
- [Scenarios \(p. 4091\)](#)

## Actions

### Create a dataset

The following code example shows how to create a Lookout for Vision dataset.

For more information, see [Creating your dataset](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Datasets:

    @staticmethod
    def create_dataset(lookoutvision_client, project_name, manifest_file,
dataset_type):
        """
        Creates a new Lookout for Vision dataset

        :param lookoutvision_client: A Lookout for Vision Boto3 client.
        :param project_name: The name of the project in which you want to
                           create a dataset.
        :param bucket: The bucket that contains the manifest file.
        :param manifest_file: The path and name of the manifest file.
        :param dataset_type: The type of the dataset (train or test).
        """
        try:
            bucket, key = manifest_file.replace("s3://", "").split("/", 1)
            logger.info("Creating %s dataset type...", dataset_type)
            dataset = {
                "GroundTruthManifest": {"S3Object": {"Bucket": bucket, "Key": key}}
            }
            response = lookoutvision_client.create_dataset(
                ProjectName=project_name,
                DatasetType=dataset_type,
                DatasetSource=dataset,
            )
            logger.info("Dataset Status: %s",
                       response["DatasetMetadata"]["Status"])
            logger.info(
                "Dataset Status Message: %s",
                response["DatasetMetadata"]["StatusMessage"],
            )
        
```

```
logger.info("Dataset Type: %s",
            response["DatasetMetadata"]["DatasetType"])

# Wait until either created or failed.
finished = False
status = ""
dataset_description = {}
while finished is False:
    dataset_description = lookoutvision_client.describe_dataset(
        ProjectName=project_name, DatasetType=dataset_type
    )
    status = dataset_description["DatasetDescription"]["Status"]

    if status == "CREATE_IN_PROGRESS":
        logger.info("Dataset creation in progress...")
        time.sleep(2)
    elif status == "CREATE_COMPLETE":
        logger.info("Dataset created.")
        finished = True
    else:
        logger.info(
            "Dataset creation failed: %s",
            dataset_description["DatasetDescription"]["StatusMessage"])
        finished = True

    if status != "CREATE_COMPLETE":
        message = dataset_description["DatasetDescription"]["StatusMessage"]
        logger.exception("Couldn't create dataset: %s", message)
        raise Exception(f"Couldn't create dataset: {message}")

except ClientError:
    logger.exception("Service error: Couldn't create dataset.")
    raise
```

- For API details, see [CreateDataset in AWS SDK for Python \(Boto3\) API Reference](#).

## Create a model

The following code example shows how to create a Lookout for Vision model.

For more information, see [Training your model](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Models:

    @staticmethod
    def create_model(
        lookoutvision_client, project_name, training_results, tag_key=None,
        tag_key_value=None):
        """
        Creates a version of a Lookout for Vision model.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project in which you want to create a
                             model.
        :param training_results: The Amazon S3 location where training results are
                               stored.
    
```

```
:param tag_key: The key for a tag to add to the model.
:param tag_key_value - A value associated with the tag_key.
:return: The model status and version.
"""
try:
    logger.info("Training model...")
    output_bucket, output_folder = training_results.replace(
        "s3://", "").split("/", 1)
    output_config = {
        "S3Location": {"Bucket": output_bucket, "Prefix": output_folder}}
    tags = []
    if tag_key is not None:
        tags = [{"Key": tag_key, "Value": tag_key_value}]

    response = lookoutvision_client.create_model(
        ProjectName=project_name, OutputConfig=output_config, Tags=tags)

    logger.info("ARN: %s", response["ModelMetadata"]["ModelArn"])
    logger.info("Version: %s", response["ModelMetadata"]["ModelVersion"])
    logger.info("Started training...")

    print("Training started. Training might take several hours to complete.")

    # Wait until training completes.
    finished = False
    status = "UNKNOWN"
    while finished is False:
        model_description = lookoutvision_client.describe_model(
            ProjectName=project_name,
            ModelVersion=response["ModelMetadata"]["ModelVersion"])
        status = model_description["ModelDescription"]["Status"]

        if status == "TRAINING":
            logger.info("Model training in progress...")
            time.sleep(600)
            continue

        if status == "TRAINED":
            logger.info("Model was successfully trained.")
        else:
            logger.info(
                "Model training failed: %s ",
                model_description["ModelDescription"]["StatusMessage"])
        finished = True
    except ClientError:
        logger.exception("Couldn't train model.")
        raise
else:
    return status, response["ModelMetadata"]["ModelVersion"]
```

- For API details, see [CreateModel in AWS SDK for Python \(Boto3\) API Reference](#).

## Create a project

The following code example shows how to create a Lookout for Vision project.

For more information, see [Creating your project](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Projects:

    @staticmethod
    def create_project(lookoutvision_client, project_name):
        """
        Creates a new Lookout for Vision project.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name for the new project.
        :return project_arn: The ARN of the new project.
        """
        try:
            logger.info("Creating project: %s", project_name)
            response = lookoutvision_client.create_project(ProjectName=project_name)
            project_arn = response["ProjectMetadata"]["ProjectArn"]
            logger.info("project ARN: %s", project_arn)
        except ClientError:
            logger.exception("Couldn't create project %s.", project_name)
            raise
        else:
            return project_arn
```

- For API details, see [CreateProject](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a dataset

The following code example shows how to delete a Lookout for Vision dataset.

For more information, see [Deleting a dataset](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Datasets:

    @staticmethod
    def delete_dataset(lookoutvision_client, project_name, dataset_type):
        """
        Deletes a Lookout for Vision dataset

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the dataset that
                             you want to delete.
        :param dataset_type: The type (train or test) of the dataset that you
                            want to delete.
        """
        try:
            logger.info(
                "Deleting the %s dataset for project %s.", dataset_type, project_name)
            lookoutvision_client.delete_dataset(
                ProjectName=project_name, DatasetType=dataset_type)
            logger.info("Dataset deleted.")
        except ClientError:
            logger.exception("Service error: Couldn't delete dataset.")
            raise
```

- For API details, see [DeleteDataset](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a model

The following code example shows how to delete a Lookout for Vision model.

For more information, see [Deleting a model](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Models:

    @staticmethod
    def delete_model(lookoutvision_client, project_name, model_version):
        """
        Deletes a Lookout for Vision model. The model must first be stopped and can't
        be in training.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the desired model.
        :param model_version: The version of the model that you want to delete.
        """
        try:
            logger.info("Deleting model: %s", model_version)
            lookoutvision_client.delete_model(
                ProjectName=project_name, ModelVersion=model_version)

            model_exists = True
            while model_exists:
                response = lookoutvision_client.list_models(ProjectName=project_name)

                model_exists = False
                for model in response["Models"]:
                    if model["ModelVersion"] == model_version:
                        model_exists = True

                if model_exists is False:
                    logger.info("Model deleted")
                else:
                    logger.info("Model is being deleted...")
                    time.sleep(2)

            logger.info("Deleted Model: %s", model_version)
        except ClientError:
            logger.exception("Couldn't delete model.")
            raise
```

- For API details, see [DeleteModel](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a project

The following code example shows how to delete a Lookout for Vision project.

For more information, see [Deleting a project](#).

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Projects:

    @staticmethod
    def delete_project(lookoutvision_client, project_name):
        """
        Deletes a Lookout for Vision Model

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that you want to delete.
        """
        try:
            logger.info("Deleting project: %s", project_name)
            response = lookoutvision_client.delete_project(ProjectName=project_name)
            logger.info("Deleted project ARN: %s ", response["ProjectArn"])
        except ClientError as err:
            logger.exception("Couldn't delete project %s.", project_name)
            raise
```

- For API details, see [DeleteProject](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a dataset

The following code example shows how to describe a Lookout for Vision dataset.

For more information, see [Viewing your dataset](#).

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Datasets:

    @staticmethod
    def describe_dataset(lookoutvision_client, project_name, dataset_type):
        """
        Gets information about a Lookout for Vision dataset.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the dataset that
                             you want to describe.
        :param dataset_type: The type (train or test) of the dataset that you want
                            to describe.
        """
        try:
            response = lookoutvision_client.describe_dataset(
                ProjectName=project_name, DatasetType=dataset_type)
            print(f"Name: {response['DatasetDescription']['ProjectName']}")
            print(f"Type: {response['DatasetDescription']['DatasetType']}")
            print(f"Status: {response['DatasetDescription']['Status']}")
            print(
                f"Message: {response['DatasetDescription']['StatusMessage']}")
            print(
                f"Images: {response['DatasetDescription']['ImageStats']['Total']}")
```

```
        print(
            f"\"Labeled: {response['DatasetDescription']['ImageStats']['Labeled']}\"")
        print(
            f"\"Normal: {response['DatasetDescription']['ImageStats']['Normal']}\"")
        print(
            f"\"Anomaly: {response['DatasetDescription']['ImageStats']['Anomaly']}\"")
    except ClientError:
        logger.exception("Service error: problem listing datasets.")
        raise
    print("Done.")
```

- For API details, see [DescribeDataset in AWS SDK for Python \(Boto3\) API Reference](#).

## Describe a model

The following code example shows how to describe a Lookout for Vision model.

For more information, see [Viewing your models](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Models:

    @staticmethod
    def describe_model(lookoutvision_client, project_name, model_version):
        """
        Shows the performance metrics for a trained model.

        :param lookoutvision_client: A Boto3 Amazon Lookout for Vision client.
        :param project_name: The name of the project that contains the desired model.
        :param model_version: The version of the model.
        """
        response = lookoutvision_client.describe_model(
            ProjectName=project_name, ModelVersion=model_version)
        model_description = response["ModelDescription"]
        print(f"\tModel version: {model_description['ModelVersion']}")
        print(f"\tARN: {model_description['ModelArn']}")
        if "Description" in model_description:
            print(f"\tDescription: {model_description['Description']}")
        print(f"\tStatus: {model_description['Status']}")
        print(f"\tMessage: {model_description['StatusMessage']}")
        print(f"\tCreated: {str(model_description['CreationTimestamp'])}")

        if model_description['Status'] in ("TRAINED", "HOSTED"):
            training_start = model_description["CreationTimestamp"]
            training_end = model_description["EvaluationEndTimestamp"]
            duration = training_end - training_start
            print(f"\tTraining duration: {duration}")

            print("\n\tPerformance metrics\n\t-----")
            print(f"\tRecall: {model_description['Performance']['Recall']}")
            print(f"\tPrecision: {model_description['Performance']['Precision']}")
            print(f"\tF1: {model_description['Performance']['F1Score']}")

            training_output_bucket = model_description["OutputConfig"]["S3Location"]
            prefix = model_description["OutputConfig"]["S3Location"]["Prefix"]
            print(f"\tTraining output: s3://{training_output_bucket}/{prefix}")
```

- For API details, see [DescribeModel](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect anomalies in an image with a trained model

The following code example shows how to detect anomalies in an image with a trained Lookout for Vision model.

For more information, see [Detecting anomalies in an image](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Inference:  
    """  
        Shows how to detect anomalies in an image using a trained Lookout for Vision model.  
    """  
  
    @staticmethod  
    def detect_anomalies(lookoutvision_client, project_name, model_version, photo):  
        """  
            Calls DetectAnomalies using the supplied project, model version, and image.  
            :param lookoutvision_client: A Lookout for Vision Boto3 client.  
            :param project: The project that contains the model that you want to use.  
            :param model_version: The version of the model that you want to use.  
            :param photo: The photo that you want to analyze.  
            :return: The DetectAnomalyResult object that contains the analysis results.  
        """  
  
        image_type = imghdr.what(photo)  
        if image_type == "jpeg":  
            content_type = "image/jpeg"  
        elif image_type == "png":  
            content_type = "image/png"  
        else:  
            logger.info("Image type not valid for %s", photo)  
            raise ValueError(  
                f"File format not valid. Supply a jpeg or png format file: {photo}")  
  
        # Get images bytes for call to detect_anomalies.  
        with open(photo, "rb") as image:  
            response = lookoutvision_client.detect_anomalies(  
                ProjectName=project_name,  
                ContentType=content_type,  
                Body=image.read(),  
                ModelVersion=model_version)  
  
        return response['DetectAnomalyResult']  
  
    @staticmethod  
    def download_from_s3(s3_resource, photo):  
        """  
            Downloads an image from an S3 bucket.  
            :param s3_resource: A Boto3 Amazon S3 resource.  
            :param photo: The Amazon S3 path of a photo to download.  
            :return: The local path to the downloaded file.  
        """
```

```

try:
    bucket, key = photo.replace("s3://", "").split("/", 1)
    local_file = os.path.basename(photo)
except ValueError:
    logger.exception("Couldn't get S3 info for %s", photo)
    raise

try:
    logger.info("Downloading %s", photo)
    s3_resource.Bucket(bucket).download_file(key, local_file)
except ClientError:
    logger.exception("Couldn't download %s from S3.", photo)
    raise

return local_file

@staticmethod
def reject_on_classification(image, prediction, confidence_limit):
    """
    Returns True if the anomaly confidence is greater than or equal to
    the supplied confidence limit.
    :param image: The name of the image file that was analyzed.
    :param prediction: The DetectAnomalyResult object returned from
    DetectAnomalies.
    :param confidence_limit: The minimum acceptable confidence (float 0 - 1).
    :return: True if the error condition indicates an anomaly, otherwise False.
    """

    reject = False

    logger.info("Checking classification for %s", image)

    if prediction['IsAnomalous'] and prediction['Confidence'] >= confidence_limit:
        reject = True
        reject_info=(f'Rejected: Anomaly confidence\n({prediction["Confidence"]:.2%}) is greater\nthan limit ({confidence_limit:.2%})')
        logger.info("%s", reject_info)

    if not reject:
        logger.info("No anomalies found.")
    return reject

@staticmethod
def reject_on_anomaly_types(image, prediction, confidence_limit,
                           anomaly_types_limit):
    """
    Checks if the number of anomaly types is greater than the anomaly types
    limit and if the prediction confidence is greater than the confidence limit.
    :param image: The name of the image file that was analyzed.
    :param prediction: The DetectAnomalyResult object returned from
    DetectAnomalies.
    :param confidence: The minimum acceptable confidence (float 0 - 1).
    :param anomaly_types_limit: The maximum number of allowable anomaly types
    (int).
    :return: True if the error condition indicates an anomaly, otherwise False.
    """

    logger.info("Checking number of anomaly types for %s",image)

    reject = False

    if prediction['IsAnomalous'] and prediction['Confidence'] >= confidence_limit:
        anomaly_types = {anomaly['Name'] for anomaly in prediction['Anomalies']\
                         if anomaly['Name'] != 'background'}

```

```

        if len(anomaly_types) > anomaly_types_limit:
            reject = True
            reject_info = (f'Rejected: Anomaly confidence
({prediction['Confidence']:.2%}) '
                           f'is greater than limit ({confidence_limit:.2%}) and '
                           f'the number of anomaly types ({len(anomaly_types)-1}) is '
                           f'greater than the limit ({anomaly_types_limit})')

            logger.info("%s", reject_info)

        if not reject:
            logger.info("No anomalies found.")
        return reject

    @staticmethod
    def reject_on_coverage(image, prediction, confidence_limit, anomaly_label,
                          coverage_limit):
        """
        Checks if the coverage area of an anomaly is greater than the coverage limit
        and if
            the prediction confidence is greater than the confidence limit.
        :param image: The name of the image file that was analyzed.
        :param prediction: The DetectAnomalyResult object returned from
        DetectAnomalies.
        :param confidence_limit: The minimum acceptable confidence (float 0-1).
        :param anomaly_label: The anomaly label for the type of anomaly that you want to
        check.
        :param coverage_limit: The maximum acceptable percentage coverage of an anomaly
        (float 0-1).
        :return: True if the error condition indicates an anomaly, otherwise False.
        """

        reject = False

        logger.info("Checking coverage for %s", image)

        if prediction['IsAnomalous'] and prediction['Confidence'] >= confidence_limit:
            for anomaly in prediction['Anomalies']:
                if (anomaly['Name'] == anomaly_label and
                    anomaly['PixelAnomaly']['TotalPercentageArea'] >
                    coverage_limit):
                    reject = True
                    reject_info=(f'Rejected: Anomaly confidence
({prediction['Confidence']:.2%}) '
                           f'is greater than limit ({confidence_limit:.2%}) and '
                           f'anomaly[{Name}] '
                           f'coverage ({anomaly['PixelAnomaly']
                           ['TotalPercentageArea']:.2%}) '
                           f'is greater than limit ({coverage_limit:.2%})')

                    logger.info("%s", reject_info)

        if not reject:
            logger.info("No anomalies found.")

        return reject

    @staticmethod
    def analyze_image(lookoutvision_client, image, config):
        """
        Analyzes an image with an Amazon Lookout for Vision model. Also
        runs a series of checks to determine if the contents of an image
        should be rejected.
        :param lookoutvision_client: A Lookout for Vision Boto3 client.
        :param image: A local image that you want to analyze.
        """

```

```
param config: Configuration information for the model and reject
limits.
"""

project = config['project']
model_version = config['model_version']
confidence_limit = config['confidence_limit']
coverage_limit = config['coverage_limit']
anomaly_types_limit = config['anomaly_types_limit']
anomaly_label = config['anomaly_label']

# Get analysis results.
print(f"Analyzing {image}.")

prediction = Inference.detect_anomalies(
    lookoutvision_client, project, model_version, image)

anomalies = []

reject = Inference.reject_on_classification(
    image, prediction, confidence_limit)

if reject:
    anomalies.append("Classification: An anomaly was found.")

reject = Inference.reject_on_coverage(
    image, prediction, confidence_limit, anomaly_label, coverage_limit)

if reject:
    anomalies.append("Coverage: Anomaly coverage too high.")

reject = Inference.reject_on_anomaly_types(
    image, prediction, confidence_limit, anomaly_types_limit)

if reject:
    anomalies.append(
        "Anomaly type count: Too many anomaly types found.")
    print()

if len(anomalies) > 0:
    print(f"Anomalies found in {image}")
    for anomaly in anomalies:
        print(f"{anomaly}")
else:
    print(f"No anomalies found in {image}")

def main():
"""
Detects anomalies in an image file.
"""
try:
    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    parser = argparse.ArgumentParser(
        description="Find anomalies with Amazon Lookout for Vision.")
    parser.add_argument(
        "image",
        help="The file that you want to analyze. Supply a local file path or a "
             "path to an S3 object.")
    parser.add_argument(
        "config", help=("The configuration JSON file to use. "
                       "See https://github.com/awsdocs/aws-doc-sdk-examples/blob/main/"))


```

```
"python/example_code/lookoutvision/README.md")  
  
args = parser.parse_args()  
  
lookoutvision_client = boto3.client("lookoutvision")  
s3_resource = boto3.resource('s3')  
  
# Get configuration information.  
with open(args.config, encoding="utf-8") as config_file:  
    config = json.load(config_file)  
  
# Download image if located in S3 bucket.  
if args.image.startswith("s3://"):  
    image = Inference.download_from_s3(s3_resource, args.image)  
else:  
    image = args.image  
  
Inference.analyze_image(lookoutvision_client, image, config)  
  
# Delete image, if downloaded from S3 bucket.  
if args.image.startswith("s3://"):  
    os.remove(image)  
  
except ClientError as err:  
    print(f"Service error: {err.response['Error']['Message']}")  
except FileNotFoundError as err:  
    print(f"The supplied file couldn't be found: {err.filename}.")  
except ValueError as err:  
    print(f"A value error occurred: {err}.")  
else:  
    print("\nSuccessfully completed analysis.")  
  
if __name__ == "__main__":  
    main()
```

- For API details, see [DetectAnomalies in AWS SDK for Python \(Boto3\) API Reference](#).

## List models

The following code example shows how to list Lookout for Vision models.

For more information, see [Viewing your models](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Models:  
  
    @staticmethod  
    def describe_models(lookoutvision_client, project_name):  
        """  
        Gets information about all models in a Lookout for Vision project.  
  
        :param lookoutvision_client: A Boto3 Lookout for Vision client.  
        :param project_name: The name of the project that you want to use.  
        """  
        try:  
            response = lookoutvision_client.list_models(ProjectName=project_name)
```

```
print("Project: " + project_name)
for model in response["Models"]:
    Models.describe_model(
        lookoutvision_client, project_name, model["ModelVersion"])
    print()
print("Done...")
except ClientError:
    logger.exception("Couldn't list models.")
    raise
```

- For API details, see [ListModels](#) in *AWS SDK for Python (Boto3) API Reference*.

## List projects

The following code example shows how to list Lookout for Vision projects.

For more information, see [Viewing your projects](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Projects:

    @staticmethod
    def list_projects(lookoutvision_client):
        """
        Lists information about the projects that are in in your AWS account
        and in the current AWS Region.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        """
        try:
            response = lookoutvision_client.list_projects()
            for project in response["Projects"]:
                print("Project: " + project["ProjectName"])
                print("\tARN: " + project["ProjectArn"])
                print("\tCreated: " + str(["CreationTimestamp"]))
                print("Datasets")
                project_description = lookoutvision_client.describe_project(
                    ProjectName=project["ProjectName"])
                if not project_description["ProjectDescription"]["Datasets"]:
                    print("\tNo datasets")
                else:
                    for dataset in project_description["ProjectDescription"][
                        "Datasets"]:
                        print(f"\t\ttype: {dataset['DatasetType']}")
                        print(f"\t\tStatus: {dataset['StatusMessage']}")

                print("Models")
                response_models = lookoutvision_client.list_models(
                    ProjectName=project["ProjectName"])
                if not response_models["Models"]:
                    print("\tNo models")
                else:
                    for model in response_models["Models"]:
                        Models.describe_model(
                            lookoutvision_client, project["ProjectName"],
                            model["ModelVersion"])
```

```
        print("-----\n")
        print("Done!")
    except ClientError:
        logger.exception("Problem listing projects.")
        raise
```

- For API details, see [ListProjects](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start a model

The following code example shows how to start a Lookout for Vision model.

For more information, see [Starting your model](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Hosting:

    @staticmethod
    def start_model(
        lookoutvision_client, project_name, model_version, min_inference_units):
        """
        Starts the hosting of a Lookout for Vision model.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the version of the
            model that you want to start hosting.
        :param model_version: The version of the model that you want to start hosting.
        :param min_inference_units: The number of inference units to use for hosting.
        """
        try:
            logger.info(
                "Starting model version %s for project %s", model_version,
                project_name)
            lookoutvision_client.start_model(
                ProjectName=project_name,
                ModelVersion=model_version,
                MinInferenceUnits=min_inference_units)
            print("Starting hosting...")

            status = ""
            finished = False

            # Wait until hosted or failed.
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name, ModelVersion=model_version)
                status = model_description["ModelDescription"]["Status"]

                if status == "STARTING_HOSTING":
                    logger.info("Host starting in progress...")
                    time.sleep(10)
                    continue

                if status == "HOSTED":
                    logger.info("Model is hosted and ready for use.")
                    finished = True
                    continue

        except ClientError:
            logger.exception("Problem starting model.")
```

```
        logger.info("Model hosting failed and the model can't be used.")
        finished = True

    if status != "HOSTED":
        logger.error("Error hosting model: %s", status)
        raise Exception(f"Error hosting model: {status}")
    except ClientError:
        logger.exception("Couldn't host model.")
        raise
```

- For API details, see [StartModel](#) in *AWS SDK for Python (Boto3) API Reference*.

## Stop a model

The following code example shows how to stop a Lookout for Vision model.

For more information, see [Stopping your model](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Hosting:

    @staticmethod
    def stop_model(lookoutvision_client, project_name, model_version):
        """
        Stops a running Lookout for Vision Model.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the version of
                             the model that you want to stop hosting.
        :param model_version: The version of the model that you want to stop hosting.
        """
        try:
            logger.info("Stopping model version %s for %s", model_version,
project_name)
            response = lookoutvision_client.stop_model(
                ProjectName=project_name, ModelVersion=model_version)
            logger.info("Stopping hosting...")

            status = response["Status"]
            finished = False

            # Wait until stopped or failed.
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name, ModelVersion=model_version)
                status = model_description["ModelDescription"]["Status"]

                if status == "STOPPING_HOSTING":
                    logger.info("Host stopping in progress...")
                    time.sleep(10)
                    continue

                if status == "TRAINED":
                    logger.info("Model is no longer hosted.")
                    finished = True
                    continue
```

```
        logger.info("Failed to stop model: %s ", status)
        finished = True

        if status != "TRAINED":
            logger.error("Error stopping model: %s", status)
            raise Exception(f"Error stopping model: {status}")
    except ClientError:
        logger.exception("Couldn't stop hosting model.")
        raise
```

- For API details, see [StopModel](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Create a manifest file

The following code example shows how to create a Lookout for Vision manifest file and upload it to Amazon S3.

For more information, see [Creating a manifest file](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Datasets:

    @staticmethod
    def create_manifest_file_s3(s3_resource, image_s3_path, manifest_s3_path):
        """
        Creates a manifest file and uploads to Amazon S3.

        :param s3_resource: A Boto3 Amazon S3 resource.
        :param image_s3_path: The Amazon S3 path to the images referenced by the
                             manifest file. The images must be in an Amazon S3 bucket
                             with the following folder structure.
                             s3://doc-example-bucket/<train or test>/
                             normal/
                             anomaly/
                             Place normal images in the normal folder and anomalous
                             images in the anomaly folder.
        :param manifest_s3_path: The Amazon S3 location in which to store the created
                               manifest file.
        """
        output_manifest_file = "temp.manifest"
        try:
            # Current date and time in manifest file format.
            dttm = datetime.now().strftime("%Y-%m-%dT%H:%M:%S.%f")

            # Get bucket and folder from image and manifest file paths.
            bucket, prefix = image_s3_path.replace("s3://", "").split("/", 1)
            if prefix[-1] != '/':
                prefix += '/'
            manifest_bucket, manifest_prefix = manifest_s3_path.replace(
                "s3://", "").split("/", 1)

            with open(output_manifest_file, "w") as mfile:
```

```
logger.info("Creating manifest file")
src_bucket = s3_resource.Bucket(bucket)

# Create JSON lines for anomalous images.
for obj in src_bucket.objects.filter(
    Prefix=prefix + "anomaly/", Delimiter="/"):
    image_path = f"s3://{src_bucket.name}/{obj.key}"
    manifest = Datasets.create_json_line(
        image_path, "anomaly", dttm)
    mfile.write(json.dumps(manifest) + "\n")

# Create json lines for normal images.
for obj in src_bucket.objects.filter(
    Prefix=prefix + "normal/", Delimiter="/"):
    image_path = f"s3://{src_bucket.name}/{obj.key}"
    manifest = Datasets.create_json_line(
        image_path, "normal", dttm)
    mfile.write(json.dumps(manifest) + "\n")

logger.info("Uploading manifest file to %s", manifest_s3_path)
s3_resource.Bucket(manifest_bucket).upload_file(
    output_manifest_file, manifest_prefix)
except ClientError:
    logger.exception("Error uploading manifest.")
    raise
except Exception:
    logger.exception("Error uploading manifest.")
    raise
else:
    logger.info("Completed manifest file creation and upload.")
finally:
    try:
        os.remove(output_manifest_file)
    except FileNotFoundError:
        pass

@staticmethod
def create_json_line(image, class_name, dttm):
    """
    Creates a single JSON line for an image.

    :param image: The S3 location for the image.
    :param class_name: The class of the image (normal or anomaly)
    :param dttm: The date and time that the JSON is created.
    """

    label = 0
    if class_name == "normal":
        label = 0
    elif class_name == "anomaly":
        label = 1
    else:
        logger.error("Unexpected label value: %s for %s", label, image)
        raise Exception(f"Unexpected label value: {label} for {image}")

    manifest = {
        "source-ref": image,
        "anomaly-label": label,
        "anomaly-label-metadata": {
            "confidence": 1,
            "job-name": "labeling-job/anomaly-label",
            "class-name": class_name,
            "human-annotated": "yes",
            "creation-date": dttm,
            "type": "groundtruth/image-classification",
        },
    }
```

```
}
```

```
return manifest
```

## Create, train, and start a model

The following code example shows how to create, train, and start a Lookout for Vision model.

### SDK for Python (Boto3)

Creates and optionally starts an Amazon Lookout for Vision model using command line arguments. The example code creates a new project, training dataset, optional test dataset, and model. After model training is completed, you can use a provided script to try your model with an image.

This example requires a set of images to train its model. You can find example circuit board images on GitHub that you can use for training and testing. For details on how to copy these images to an Amazon Simple Storage Service (Amazon S3) bucket, see [Prepare example images](#).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Lookout for Vision

## Find a project with a specific tag

The following code example shows how to find a Lookout for Vision project with a specific tag.

For more information, see [Tagging models](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import logging
import argparse
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def find_tag(tags, key, value):
    """
    Finds a tag in the supplied list of tags.

    :param tags: A list of tags associated with a Lookout for Vision model.
    :param key: The tag to search for.
    :param value: The tag key value to search for.
    :return: True if the tag value exists, otherwise False.
    """

    found = False
    for tag in tags:
        if key == tag["Key"]:
            logger.info("\t\tMatch found for tag: %s value: %s.", key, value)
            found = True
```

```
        break
    return found

def find_tag_in_projects(lookoutvision_client, key, value):
    """
    Finds Lookout for Vision models tagged with the supplied key and value.

    :param lookoutvision_client: A Boto3 Lookout for Vision client.
    :param key: The tag key to find.
    :param value: The value of the tag that you want to find.
    :return: A list of matching model versions (and model projects) that were found.
    """
    try:
        found_tags = []
        found = False
        projects = lookoutvision_client.list_projects()
        # Iterate through each project and models within a project.
        for project in projects["Projects"]:
            logger.info("Searching project: %s ...", project["ProjectName"])

            response_models = lookoutvision_client.list_models(
                ProjectName=project["ProjectName"])
            for model in response_models["Models"]:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project["ProjectName"],
                    ModelVersion=model["ModelVersion"])
                tags = lookoutvision_client.list_tags_for_resource(
                    ResourceArn=model_description["ModelDescription"]["ModelArn"])

                logger.info(
                    "\t\tSearching model: %s for tag: %s value: %s.",
                    model_description["ModelDescription"]["ModelArn"], key, value)
                if find_tag(tags["Tags"], key, value) is True:
                    found = True
                    logger.info(
                        "\t\tMATCH: Project: %s: model version %s",
                        project["ProjectName"],
                        model_description["ModelDescription"]["ModelVersion"])
                    found_tags.append({
                        "Project": project["ProjectName"],
                        "ModelVersion":
                            model_description["ModelDescription"]["ModelVersion"]})
    except ClientError:
        logger.exception("Problem finding tags.")
        raise
    else:
        return found_tags

def main():
    logging.basicConfig(level=logging.INFO, format"%(levelname)s: %(message)s")
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
    parser.add_argument("tag", help="The tag that you want to find.")
    parser.add_argument("value", help="The tag value that you want to find.")
    args = parser.parse_args()
    key = args.tag
    value = args.value
    lookoutvision_client = boto3.client("lookoutvision")

    print(f"Searching your models for tag: {key} with value: {value}.")
    tagged_models = find_tag_in_projects(lookoutvision_client, key, value)

    print("Matched models\n-----")
```

```
if len(tagged_models) > 0:
    for model in tagged_models:
        print(f"Project: {model['Project']}. model version:
{model['ModelVersion']}")"
    else:
        print("No matches found.")

if __name__ == "__main__":
    main()
```

### List models that are currently hosted

The following code example shows how to list Lookout for Vision models that are currently hosted.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class Hosting:

    @staticmethod
    def list_hosted(lookoutvision_client):
        """
        Displays a list of models in your account that are currently hosted.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        """
        try:
            response = lookoutvision_client.list_projects()
            hosted = 0
            print("Hosted models\n-----")

            for project in response["Projects"]:
                response_models = lookoutvision_client.list_models(
                    ProjectName=project["ProjectName"])

                for model in response_models["Models"]:
                    model_description = lookoutvision_client.describe_model(
                        ProjectName=project["ProjectName"],
                        ModelVersion=model["ModelVersion"])

                    if model_description["ModelDescription"]["Status"] == "HOSTED":
                        print(
                            f"Project: {project['ProjectName']} Model version: "
                            f"{model['ModelVersion']}")
                        hosted += 1
            print(f"{hosted} model(s) hosted")
        except ClientError:
            logger.exception("Problem listing hosted models.")
            raise
```

## Organizations examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with AWS Organizations.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Topics

- [Actions \(p. 4096\)](#)

## Actions

### Attach a policy to a target

The following code example shows how to attach an Organizations policy to a target.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def attach_policy(policy_id, target_id, orgs_client):
    """
    Attaches a policy to a target. The target is an organization root, account, or
    organizational unit.

    :param policy_id: The ID of the policy to attach.
    :param target_id: The ID of the resources to attach the policy to.
    :param orgs_client: The Boto3 Organizations client.
    """
    try:
        orgs_client.attach_policy(PolicyId=policy_id, TargetId=target_id)
        logger.info("Attached policy %s to target %s.", policy_id, target_id)
    except ClientError:
        logger.exception(
            "Couldn't attach policy %s to target %s.", policy_id, target_id)
        raise
```

- For API details, see [AttachPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

### Create a policy

The following code example shows how to create an Organizations policy.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_policy(name, description, content, policy_type, orgs_client):
    """
    Creates a policy.

    :param name: The name of the policy.
    :param description: The description of the policy.
    :param content: The policy content as a dict. This is converted to JSON before
                   it is sent to AWS. The specific format depends on the policy type.
    :param policy_type: The type of the policy.
    :param orgs_client: The Boto3 Organizations client.
```

```
:return: The newly created policy.  
"""  
try:  
    response = orgs_client.create_policy(  
        Name=name, Description=description, Content=json.dumps(content),  
        Type=policy_type)  
    policy = response['Policy']  
    logger.info("Created policy %s.", name)  
except ClientError:  
    logger.exception("Couldn't create policy %s.", name)  
    raise  
else:  
    return policy
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a policy

The following code example shows how to delete an Organizations policy.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_policy(policy_id, orgs_client):  
    """  
    Deletes a policy.  
  
    :param policy_id: The ID of the policy to delete.  
    :param orgs_client: The Boto3 Organizations client.  
    """  
    try:  
        orgs_client.delete_policy(PolicyId=policy_id)  
        logger.info("Deleted policy %s.", policy_id)  
    except ClientError:  
        logger.exception(  
            "Couldn't delete policy %s.", policy_id)  
        raise
```

- For API details, see [DeletePolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a policy

The following code example shows how to describe an Organizations policy.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def describe_policy(policy_id, orgs_client):  
    """  
    Describes a policy.  
  
    :param policy_id: The ID of the policy to describe.  
    """
```

```
:param orgs_client: The Boto3 Organizations client.  
:return: The description of the policy.  
"""  
try:  
    response = orgs_client.describe_policy(PolicyId=policy_id)  
    policy = response['Policy']  
    logger.info("Got policy %s.", policy_id)  
except ClientError:  
    logger.exception("Couldn't get policy %s.", policy_id)  
    raise  
else:  
    return policy
```

- For API details, see [DescribePolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detach a policy from a target

The following code example shows how to detach an Organizations policy from a target.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def detach_policy(policy_id, target_id, orgs_client):  
    """  
    Detaches a policy from a target.  
  
    :param policy_id: The ID of the policy to detach.  
    :param target_id: The ID of the resource where the policy is currently attached.  
    :param orgs_client: The Boto3 Organizations client.  
    """  
    try:  
        orgs_client.detach_policy(PolicyId=policy_id, TargetId=target_id)  
        logger.info("Detached policy %s from target %s.", policy_id, target_id)  
    except ClientError:  
        logger.exception(  
            "Couldn't detach policy %s from target %s.", policy_id, target_id)  
        raise
```

- For API details, see [DetachPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## List policies

The following code example shows how to list Organizations policies.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_policies(policy_filter, orgs_client):  
    """  
    Lists the policies for the account, limited to the specified filter.  
  
    :param policy_filter: The kind of policies to return.  
    """
```

```
:param orgs_client: The Boto3 Organizations client.  
:return: The list of policies found.  
"""  
try:  
    response = orgs_client.list_policies(Filter=policy_filter)  
    policies = response['Policies']  
    logger.info("Found %s %s policies.", len(policies), policy_filter)  
except ClientError:  
    logger.exception("Couldn't get %s policies.", policy_filter)  
    raise  
else:  
    return policies
```

- For API details, see [ListPolicies](#) in *AWS SDK for Python (Boto3) API Reference*.

## Amazon Pinpoint examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Pinpoint.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4099\)](#)

## Actions

### Send email and text messages

The following code example shows how to send email and text messages with Amazon Pinpoint.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Send an email message.

```
import logging  
import boto3  
from botocore.exceptions import ClientError  
  
logger = logging.getLogger(__name__)  
  
def send_email_message(  
    pinpoint_client, app_id, sender, to_addresses, char_set, subject,  
    html_message, text_message):  
    """  
        Sends an email message with HTML and plain text versions.  
    :param pinpoint_client: A Boto3 Pinpoint client.  
    :param app_id: The Amazon Pinpoint project ID to use when you send this message.  
    :param sender: The "From" address. This address must be verified in  
                  Amazon Pinpoint in the AWS Region you're using to send email.  
    """
```

```

:param to_addresses: The addresses on the "To" line. If your Amazon Pinpoint
account
    is in the sandbox, these addresses must be verified.
:param char_set: The character encoding to use for the subject line and message
body of the email.
:param subject: The subject line of the email.
:param html_message: The body of the email for recipients whose email clients can
display HTML content.
:param text_message: The body of the email for recipients whose email clients
don't support HTML content.
:return: A dict of to_addresses and their message IDs.
"""

try:
    response = pinpoint_client.send_messages(
        ApplicationId=app_id,
        MessageRequest={
            'Addresses': {
                to_address: {'ChannelType': 'EMAIL'} for to_address in to_addresses
            },
            'MessageConfiguration': {
                'EmailMessage': {
                    'FromAddress': sender,
                    'SimpleEmail': {
                        'Subject': {'Charset': char_set, 'Data': subject},
                        'HtmlPart': {'Charset': char_set, 'Data': html_message},
                        'TextPart': {'Charset': char_set, 'Data':
text_message}}}}})
    except ClientError:
        logger.exception("Couldn't send email.")
        raise
else:
    return {
        to_address: message['MessageId'] for
        to_address, message in response['MessageResponse']['Result'].items()
    }

def main():
    app_id = "ce796be37f32f178af652b26eexample"
    sender = "sender@example.com"
    to_address = "recipient@example.com"
    char_set = "UTF-8"
    subject = "Amazon Pinpoint Test (SDK for Python (Boto3))"
    text_message = """Amazon Pinpoint Test (SDK for Python)

This email was sent with Amazon Pinpoint using the AWS SDK for Python (Boto3).
For more information, see https://aws.amazon.com/sdk-for-python/"""

    html_message = """<html>
<head></head>
<body>
    <h1>Amazon Pinpoint Test (SDK for Python (Boto3))</h1>
    <p>This email was sent with
        <a href='https://aws.amazon.com/pinpoint/'>Amazon Pinpoint</a> using the
        <a href='https://aws.amazon.com/sdk-for-python/'>
            AWS SDK for Python (Boto3)</a>.</p>
</body>
</html>
"""

    print("Sending email.")
    message_ids = send_email_message(
        boto3.client('pinpoint'), app_id, sender, [to_address], char_set, subject,
        html_message, text_message)
    print(f"Message sent! Message IDs: {message_ids}")

```

```
if __name__ == '__main__':
    main()
```

Send an SMS message.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def send_sms_message(
        pinpoint_client, app_id, origination_number, destination_number, message,
        message_type):
    """
    Sends an SMS message with Amazon Pinpoint.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param app_id: The Amazon Pinpoint project/application ID to use when you send
                   this message. The SMS channel must be enabled for the project or
                   application.
    :param destination_number: The recipient's phone number in E.164 format.
    :param origination_number: The phone number to send the message from. This phone
                               number must be associated with your Amazon Pinpoint
                               account and be in E.164 format.
    :param message: The content of the SMS message.
    :param message_type: The type of SMS message that you want to send. If you send
                         time-sensitive content, specify TRANSACTIONAL. If you send
                         marketing-related content, specify PROMOTIONAL.
    :return: The ID of the message.
    """
    try:
        response = pinpoint_client.send_messages(
            ApplicationId=app_id,
            MessageRequest={
                'Addresses': {destination_number: {'ChannelType': 'SMS'}},
                'MessageConfiguration': {
                    'SMSMessage': {
                        'Body': message,
                        'MessageType': message_type,
                        'OriginationNumber': origination_number}}})
    except ClientError:
        logger.exception("Couldn't send message.")
        raise
    else:
        return response['MessageResponse']['Result'][destination_number]['MessageId']

def main():
    app_id = "ce796be37f32f178af652b26eexample"
    origination_number = "+12065550199"
    destination_number = "+14255550142"
    message = (
        "This is a sample message sent from Amazon Pinpoint by using the AWS SDK for "
        "Python (Boto 3.)")
    message_type = "TRANSACTIONAL"

    print("Sending SMS message.")
    message_id = send_sms_message(
        boto3.client('pinpoint'), app_id, origination_number, destination_number,
        message, message_type)
```

```
    print(f"Message sent! Message ID: {message_id}.")  
  
if __name__ == '__main__':  
    main()
```

- For API details, see [SendMessages](#) in *AWS SDK for Python (Boto3) API Reference*.

## Send templated email and text messages

The following code example shows how to send templated email and text messages with Amazon Pinpoint.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Send an email message with an existing email template.

```
import logging  
import boto3  
from botocore.exceptions import ClientError  
  
logger = logging.getLogger(__name__)  
  
def send_templated_email_message(  
    pinpoint_client, project_id, sender, to_addresses, template_name,  
    template_version):  
    """  
    Sends an email message with HTML and plain text versions.  
  
    :param pinpoint_client: A Boto3 Pinpoint client.  
    :param project_id: The Amazon Pinpoint project ID to use when you send this  
        message.  
    :param sender: The "From" address. This address must be verified in  
        Amazon Pinpoint in the AWS Region you're using to send email.  
    :param to_addresses: The addresses on the "To" line. If your Amazon Pinpoint  
        account is in the sandbox, these addresses must be verified.  
    :param template_name: The name of the email template to use when sending the  
        message.  
    :param template_version: The version number of the message template.  
  
    :return: A dict of to_addresses and their message IDs.  
    """  
    try:  
        response = pinpoint_client.send_messages(  
            ApplicationId=project_id,  
            MessageRequest={  
                'Addresses': {  
                    'to_address': {  
                        'ChannelType': 'EMAIL'  
                    } for to_address in to_addresses  
                },  
                'MessageConfiguration': {  
                    'EmailMessage': {'FromAddress': sender}  
                },  
                'TemplateConfiguration': {  
                    'EmailTemplate': {  
                        'Name': template_name,  
                    }  
                }  
            }  
        )  
        return response['Messages']  
    except ClientError as e:  
        logger.error(e)  
        raise e
```

```
        'Version': template_version
    }
}
)
except ClientError:
    logger.exception("Couldn't send email.")
    raise
else:
    return {
        to_address: message['MessageId'] for
        to_address, message in response['MessageResponse']['Result'].items()
    }

def main():
    project_id = "296b04b342374fc6b661bf494example"
    sender = "sender@example.com"
    to_addresses = ["recipient@example.com"]
    template_name = "My_Email_Template"
    template_version = "1"

    print("Sending email.")
    message_ids = send_templated_email_message(
        boto3.client('pinpoint'), project_id, sender, to_addresses, template_name,
        template_version)
    print(f"Message sent! Message IDs: {message_ids}")

if __name__ == '__main__':
    main()
```

Send a text message with an existing SMS template.

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def send_templated_sms_message(
    pinpoint_client,
    project_id,
    destination_number,
    message_type,
    origination_number,
    template_name,
    template_version):
    """
    Sends an SMS message to a specific phone number using a pre-defined template.

    :param pinpoint_client: A Boto3 Pinpoint client.
    :param project_id: An Amazon Pinpoint project (application) ID.
    :param destination_number: The phone number to send the message to.
    :param message_type: The type of SMS message (promotional or transactional).
    :param origination_number: The phone number that the message is sent from.
    :param template_name: The name of the SMS template to use when sending the message.
    :param template_version: The version number of the message template.

    :return The ID of the message.
    """
    try:
        response = pinpoint_client.send_messages(
```

```
        ApplicationId=project_id,
        MessageRequest={
            'Addresses': {
                'destination_number: {
                    'ChannelType': 'SMS'
                }
            },
            'MessageConfiguration': {
                'SMSMessage': {
                    'MessageType': message_type,
                    'OriginationNumber': origination_number
                }
            },
            'TemplateConfiguration': {
                'SMSTemplate': {
                    'Name': template_name,
                    'Version': template_version
                }
            }
        }
    )

except ClientError:
    logger.exception("Couldn't send message.")
    raise
else:
    return response['MessageResponse']['Result'][destination_number]['MessageId']

def main():
    region = "us-east-1"
    origination_number = "+18555550001"
    destination_number = "+14255550142"
    project_id = "7353f53e6885409fa32d07cedexample"
    message_type = "TRANSACTIONAL"
    template_name = "My_SMS_Template"
    template_version = "1"
    message_id = send_templated_sms_message(
        boto3.client('pinpoint', region_name=region), project_id,
        destination_number, message_type, origination_number, template_name,
        template_version)
    print(f"Message sent! Message ID: {message_id}.")

if __name__ == '__main__':
    main()
```

- For API details, see [SendMessages](#) in *AWS SDK for Python (Boto3) API Reference*.

## Amazon Pinpoint SMS and Voice API examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Pinpoint SMS and Voice API.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4105\)](#)

## Actions

### Send a voice message with Amazon Pinpoint SMS and Voice API

The following code example shows how to send a voice message with Amazon Pinpoint SMS and Voice API.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def send_voice_message(
    sms_voice_client, origination_number, caller_id, destination_number,
    language_code, voice_id, ssml_message):
    """
    Sends a voice message using speech synthesis provided by Amazon Polly.

    :param sms_voice_client: A Boto3 PinpointSMSVoice client.
    :param origination_number: The phone number that the message is sent from.
        The phone number must be associated with your Amazon Pinpoint account and be in E.164 format.
    :param caller_id: The phone number that you want to appear on the recipient's device. The phone number must be associated with your Amazon Pinpoint account and be in E.164 format.
    :param destination_number: The recipient's phone number. Specify the phone number in E.164 format.
    :param language_code: The language to use when sending the message.
    :param voice_id: The Amazon Polly voice that you want to use to send the message.
    :param ssml_message: The content of the message. This example uses SSML to control certain aspects of the message, such as the volume and the speech rate. The message must not contain line breaks.
    :return: The ID of the message.
    """
    try:
        response = sms_voice_client.send_voice_message(
            DestinationPhoneNumber=destination_number,
            OriginationPhoneNumber=origination_number,
            CallerId=caller_id,
            Content={
                'SSMLMessage': {
                    'LanguageCode': language_code,
                    'VoiceId': voice_id,
                    'Text': ssml_message}})
    except ClientError:
        logger.exception(
            "Couldn't send message from %s to %s.", origination_number,
            destination_number)
        raise
    else:
        return response['MessageId']
```

```
def main():
    origination_number = "+12065550110"
    caller_id = "+12065550199"
    destination_number = "+12065550142"
    language_code = "en-US"
    voice_id = "Matthew"
    ssml_message = (
        "<speak>"
        "This is a test message sent from <emphasis>Amazon Pinpoint</emphasis> "
        "<using the <break strength='weak' />AWS SDK for Python (Boto3). "
        "<><amazon:effect phonation='soft'>Thank you for listening."
        "</amazon:effect>"
        "</speak>")
    print(f"Sending voice message from {origination_number} to {destination_number}.")
    message_id = send_voice_message(
        boto3.client('pinpoint-sms-voice'), origination_number, caller_id,
        destination_number, language_code, voice_id, ssml_message)
    print(f"Message sent!\nMessage ID: {message_id}")

if __name__ == '__main__':
    main()
```

- For API details, see [SendVoiceMessage](#) in *AWS SDK for Python (Boto3) API Reference*.

## Amazon Polly examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Polly.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4106\)](#)
- [Scenarios \(p. 4112\)](#)

## Actions

### Get a lexicon

The following code example shows how to get an Amazon Polly lexicon.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""
    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3) resource.
        """
```

```
    self.polly_client = polly_client
    self.s3_resource = s3_resource
    self.voice_metadata = None

    def get_lexicon(self, name):
        """
        Gets metadata and contents of an existing lexicon.

        :param name: The name of the lexicon to retrieve.
        :return: The retrieved lexicon.
        """
        try:
            response = self.polly_client.get_lexicon(Name=name)
            logger.info("Got lexicon %s.", name)
        except ClientError:
            logger.exception("Couldn't get lexicon %s.", name)
            raise
        else:
            return response
```

- For API details, see [GetLexicon](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get data about a speech synthesis task

The following code example shows how to get data about an existing asynchronous Amazon Polly speech synthesis task.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""
    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3) resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None

    def get_speech_synthesis_task(self, task_id):
        """
        Gets metadata about an asynchronous speech synthesis task, such as its status.

        :param task_id: The ID of the task to retrieve.
        :return: Metadata about the task.
        """
        try:
            response = self.polly_client.get_speech_synthesis_task(TaskId=task_id)
            task = response['SynthesisTask']
            logger.info("Got synthesis task. Status is %s.", task['TaskStatus'])
        except ClientError:
            logger.exception("Couldn't get synthesis task %s.", task_id)
            raise
        else:
            return task
```

- For API details, see [GetSpeechSynthesisTask](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get voices available for synthesis

The following code example shows how to get Amazon Polly voices available for synthesis.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PollyWrapper:  
    """Encapsulates Amazon Polly functions."""  
    def __init__(self, polly_client, s3_resource):  
        """  
        :param polly_client: A Boto3 Amazon Polly client.  
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3) resource.  
        """  
        self.polly_client = polly_client  
        self.s3_resource = s3_resource  
        self.voice_metadata = None  
  
    def describe.voices(self):  
        """  
        Gets metadata about available voices.  
  
        :return: The list of voice metadata.  
        """  
        try:  
            response = self.polly_client.describe.voices()  
            self.voice_metadata = response['Voices']  
            logger.info("Got metadata about %s voices.", len(self.voice_metadata))  
        except ClientError:  
            logger.exception("Couldn't get voice metadata.")  
            raise  
        else:  
            return self.voice_metadata
```

- For API details, see [DescribeVoices](#) in *AWS SDK for Python (Boto3) API Reference*.

## List pronunciation lexicons

The following code example shows how to list Amazon Polly pronunciation lexicons.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PollyWrapper:  
    """Encapsulates Amazon Polly functions."""  
    def __init__(self, polly_client, s3_resource):  
        """  
        :param polly_client: A Boto3 Amazon Polly client.  
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3) resource.  
        """  
        self.polly_client = polly_client
```

```
self.s3_resource = s3_resource
self.voice_metadata = None

def list_lexicons(self):
    """
    Lists lexicons in the current account.

    :return: The list of lexicons.
    """
    try:
        response = self.polly_client.list_lexicons()
        lexicons = response['Lexicons']
        logger.info("Got %s lexicons.", len(lexicons))
    except ClientError:
        logger.exception("Couldn't get %s.", )
        raise
    else:
        return lexicons
```

- For API details, see [ListLexicons](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start a speech synthesis task

The following code example shows how to start an asynchronous Amazon Polly speech synthesis task.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PollyWrapper:
    """Encapsulates Amazon Polly functions."""
    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3) resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None

    def do_synthesis_task(
            self, text, engine, voice, audio_format, s3_bucket, lang_code=None,
            include_visemes=False, wait_callback=None):
        """
        Start an asynchronous task to synthesize speech or speech marks, wait for
        the task to complete, retrieve the output from Amazon S3, and return the
        data.

        An asynchronous task is required when the text is too long for near-real time
        synthesis.

        :param text: The text to synthesize.
        :param engine: The kind of engine used. Can be standard or neural.
        :param voice: The ID of the voice to use.
        :param audio_format: The audio format to return for synthesized speech. When
                            speech marks are synthesized, the output format is JSON.
        :param s3_bucket: The name of an existing Amazon S3 bucket that you have
                          write access to. Synthesis output is written to this bucket.
        :param lang_code: The language code of the voice to use. This has an effect
                          only when a bilingual voice is selected.
        """
        pass
```

```

:param include_visemes: When True, a second request is made to Amazon Polly
                      to synthesize a list of visemes, using the specified
                      text and voice. A viseme represents the visual position
                      of the face and mouth when saying part of a word.
:param wait_callback: A callback function that is called periodically during
                      task processing, to give the caller an opportunity to
                      take action, such as to display status.
:return: The audio stream that contains the synthesized speech and a list
        of visemes that are associated with the speech audio.
"""
try:
    kwargs = {
        'Engine': engine,
        'OutputFormat': audio_format,
        'OutputS3BucketName': s3_bucket,
        'Text': text,
        'VoiceId': voice}
    if lang_code is not None:
        kwargs['LanguageCode'] = lang_code
    response = self.polly_client.start_speech_synthesis_task(**kwargs)
    speech_task = response['SynthesisTask']
    logger.info("Started speech synthesis task %s.", speech_task['TaskId'])

    viseme_task = None
    if include_visemes:
        kwargs['OutputFormat'] = 'json'
        kwargs['SpeechMarkTypes'] = ['viseme']
        response = self.polly_client.start_speech_synthesis_task(**kwargs)
        viseme_task = response['SynthesisTask']
        logger.info("Started viseme synthesis task %s.", viseme_task['TaskId'])
    except ClientError:
        logger.exception("Couldn't start synthesis task.")
        raise
    else:
        bucket = self.s3_resource.Bucket(s3_bucket)
        audio_stream = self._wait_for_task(
            10, speech_task['TaskId'], 'speech', wait_callback, bucket)

        visemes = None
        if include_visemes:
            viseme_data = self._wait_for_task(
                10, viseme_task['TaskId'], 'viseme', wait_callback, bucket)
            visemes = [json.loads(v) for v in
                       viseme_data.read().decode().split() if v]

return audio_stream, visemes

```

- For API details, see [StartSpeechSynthesisTask](#) in *AWS SDK for Python (Boto3) API Reference*.

## Store a pronunciation lexicon

The following code example shows how to store an Amazon Polly pronunciation lexicon.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class PollyWrapper:
    """Encapsulates Amazon Polly functions."""
    def __init__(self, polly_client, s3_resource):

```

```
"""
:param polly_client: A Boto3 Amazon Polly client.
:param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3) resource.
"""
self.polly_client = polly_client
self.s3_resource = s3_resource
self.voice_metadata = None

def create_lexicon(self, name, content):
    """
    Creates a lexicon with the specified content. A lexicon contains custom
    pronunciations.

    :param name: The name of the lexicon.
    :param content: The content of the lexicon.
    """
    try:
        self.polly_client.put_lexicon(Name=name, Content=content)
        logger.info("Created lexicon %s.", name)
    except ClientError:
        logger.exception("Couldn't create lexicon %s.")
        raise
```

- For API details, see [PutLexicon](#) in *AWS SDK for Python (Boto3) API Reference*.

## Synthesize speech from text

The following code example shows how to synthesize speech from text with Amazon Polly.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class PollyWrapper:
    """
    Encapsulates Amazon Polly functions.
    """
    def __init__(self, polly_client, s3_resource):
        """
        :param polly_client: A Boto3 Amazon Polly client.
        :param s3_resource: A Boto3 Amazon Simple Storage Service (Amazon S3) resource.
        """
        self.polly_client = polly_client
        self.s3_resource = s3_resource
        self.voice_metadata = None

    def synthesize(
            self, text, engine, voice, audio_format, lang_code=None,
            include_visemes=False):
        """
        Synthesizes speech or speech marks from text, using the specified voice.

        :param text: The text to synthesize.
        :param engine: The kind of engine used. Can be standard or neural.
        :param voice: The ID of the voice to use.
        :param audio_format: The audio format to return for synthesized speech. When
                             speech marks are synthesized, the output format is JSON.
        :param lang_code: The language code of the voice to use. This has an effect
                         only when a bilingual voice is selected.
        :param include_visemes: When True, a second request is made to Amazon Polly
                               to synthesize a list of visemes, using the specified
                               text and voice. A viseme represents the visual position
        """

```

```
    of the face and mouth when saying part of a word.
:return: The audio stream that contains the synthesized speech and a list
        of visemes that are associated with the speech audio.
"""
try:
    kwargs = {
        'Engine': engine,
        'OutputFormat': audio_format,
        'Text': text,
        'VoiceId': voice}
    if lang_code is not None:
        kwargs['LanguageCode'] = lang_code
    response = self.polly_client.synthesize_speech(**kwargs)
    audio_stream = response['AudioStream']
    logger.info("Got audio stream spoken by %s.", voice)
    visemes = None
    if include_visemes:
        kwargs['OutputFormat'] = 'json'
        kwargs['SpeechMarkTypes'] = ['viseme']
        response = self.polly_client.synthesize_speech(**kwargs)
        visemes = [json.loads(v) for v in
                   response['AudioStream'].read().decode().split() if v]
        logger.info("Got %s visemes.", len(visemes))
    except ClientError:
        logger.exception("Couldn't get audio stream.")
        raise
else:
    return audio_stream, visemes
```

- For API details, see [SynthesizeSpeech](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Create a lip-sync application

The following code example shows how to create a lip-sync application with Amazon Polly.

#### SDK for Python (Boto3)

Shows how to use Amazon Polly and Tkinter to create a lip-sync application that displays an animated face speaking along with the speech synthesized by Amazon Polly. Lip-sync is accomplished by requesting a list of visemes from Amazon Polly that match up with the synthesized speech.

- Get voice metadata from Amazon Polly and display it in a Tkinter application.
- Get synthesized speech audio and matching viseme speech marks from Amazon Polly.
- Play the audio with synchronized mouth movements in an animated face.
- Submit asynchronous synthesis tasks for long texts and retrieve the output from an Amazon Simple Storage Service (Amazon S3) bucket.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon Polly

## Amazon RDS examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Relational Database Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4113\)](#)
- [Scenarios \(p. 4123\)](#)

## Actions

### Create a DB instance

The following code example shows how to create an Amazon RDS DB instance and wait for it to become available.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:  
    """Encapsulates Amazon RDS DB instance actions."""  
    def __init__(self, rds_client):  
        """  
        :param rds_client: A Boto3 Amazon RDS client.  
        """  
        self.rds_client = rds_client  
  
    @classmethod  
    def from_client(cls):  
        """  
        Instantiates this class from a Boto3 client.  
        """  
        rds_client = boto3.client('rds')  
        return cls(rds_client)  
  
    def create_db_instance(  
        self, db_name, instance_id, parameter_group_name, db_engine,  
        db_engine_version,  
        instance_class, storage_type, allocated_storage, admin_name,  
        admin_password):  
        """  
        Creates a DB instance.  
  
        :param db_name: The name of the database that is created in the DB instance.  
        :param instance_id: The ID to give the newly created DB instance.  
        :param parameter_group_name: A parameter group to associate with the DB  
        instance.  
        :param db_engine: The database engine of a database to create in the DB  
        instance.  
        :param db_engine_version: The engine version for the created database.  
        :param instance_class: The DB instance class for the newly created DB instance.  
        :param storage_type: The storage type of the DB instance.  
        :param allocated_storage: The amount of storage allocated on the DB instance,  
        in Gibs.  
        :param admin_name: The name of the admin user for the created database.  
        :param admin_password: The admin password for the created database.  
        :return: Data about the newly created DB instance.  
        """
```

```
try:
    response = self.rds_client.create_db_instance(
        DBName=db_name,
        DBInstanceIdentifier=instance_id,
        DBParameterGroupName=parameter_group_name,
        Engine=db_engine,
        EngineVersion=db_engine_version,
        DBInstanceClass=instance_class,
        StorageType=storage_type,
        AllocatedStorage=allocated_storage,
        MasterUsername=admin_name,
        MasterUserPassword=admin_password)
    db_inst = response['DBInstance']
except ClientError as err:
    logger.error(
        "Couldn't create DB instance %s. Here's why: %s: %s",
        instance_id,
        err.response['Error']['Code'],
        err.response['Error']['Message'])
    raise
else:
    return db_inst
```

- For API details, see [CreateDBInstance](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a DB parameter group

The following code example shows how to create an Amazon RDS DB parameter group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def create_parameter_group(self, parameter_group_name, parameter_group_family,
                               description):
        """
        Creates a DB parameter group that is based on the specified parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter group.
        :param parameter_group_family: The family that is used as the basis of the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
```

```
        response = self.rds_client.create_db_parameter_group(
            DBParameterGroupName=parameter_group_name,
            DBParameterGroupFamily=parameter_group_family,
            Description=description)
    except ClientError as err:
        logger.error(
            "Couldn't create parameter group %s. Here's why: %s: %s",
            parameter_group_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response
```

- For API details, see [CreateDBParameterGroup in AWS SDK for Python \(Boto3\) API Reference](#).

## Create a snapshot of a DB instance

The following code example shows how to create a snapshot of an Amazon RDS DB instance.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def create_snapshot(self, snapshot_id, instance_id):
        """
        Creates a snapshot of a DB instance.

        :param snapshot_id: The ID to give the created snapshot.
        :param instance_id: The ID of the DB instance to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_snapshot(
                DBSnapshotIdentifier=snapshot_id, DBInstanceIdentifier=instance_id)
            snapshot = response['DBSnapshot']
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s", instance_id,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return snapshot
```

- For API details, see [CreateDBSnapshot](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a DB instance

The following code example shows how to delete an Amazon RDS DB instance.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:  
    """Encapsulates Amazon RDS DB instance actions."""  
    def __init__(self, rds_client):  
        """  
        :param rds_client: A Boto3 Amazon RDS client.  
        """  
        self.rds_client = rds_client  
  
    @classmethod  
    def from_client(cls):  
        """  
        Instantiates this class from a Boto3 client.  
        """  
        rds_client = boto3.client('rds')  
        return cls(rds_client)  
  
    def delete_db_instance(self, instance_id):  
        """  
        Deletes a DB instance.  
  
        :param instance_id: The ID of the DB instance to delete.  
        :return: Data about the deleted DB instance.  
        """  
        try:  
            response = self.rds_client.delete_db_instance(  
                DBInstanceIdentifier=instance_id, SkipFinalSnapshot=True,  
                DeleteAutomatedBackups=True)  
            db_inst = response['DBInstance']  
        except ClientError as err:  
            logger.error(  
                "Couldn't delete DB instance %s. Here's why: %s: %s", instance_id,  
                err.response['Error']['Code'], err.response['Error']['Message'])  
            raise  
        else:  
            return db_inst
```

- For API details, see [DeleteDBInstance](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a DB parameter group

The following code example shows how to delete an Amazon RDS DB parameter group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            self.rds_client.delete_db_parameter_group(
                DBParameterGroupName=parameter_group_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

- For API details, see [DeleteDBParameterGroup in AWS SDK for Python \(Boto3\) API Reference](#).

## Describe DB instances

The following code example shows how to describe Amazon RDS DB instances.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)
```

```
def get_db_instance(self, instance_id):
    """
    Gets data about a DB instance.

    :param instance_id: The ID of the DB instance to retrieve.
    :return: The retrieved DB instance.
    """
    try:
        response = self.rds_client.describe_db_instances(
            DBInstanceIdentifier=instance_id)
        db_inst = response['DBInstances'][0]
    except ClientError as err:
        if err.response['Error']['Code'] == 'DBInstanceNotFound':
            logger.info("Instance %s does not exist.", instance_id)
        else:
            logger.error(
                "Couldn't get DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
    else:
        return db_inst
```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe DB parameter groups

The following code example shows how to describe Amazon RDS DB parameter groups.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name)
```

```
        parameter_group = response['DBParameterGroups'][0]
    except ClientError as err:
        if err.response['Error']['Code'] == 'DBParameterGroupNotFound':
            logger.info("Parameter group %s does not exist.", parameter_group_name)
        else:
            logger.error(
                "Couldn't get parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
    else:
        return parameter_group
```

- For API details, see [DescribeDBParameterGroups](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe database engine versions

The following code example shows how to describe Amazon RDS database engine versions.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned list of
                                       engine versions to those that are compatible
                                       with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {'Engine': engine}
            if parameter_group_family is not None:
                kwargs['DBParameterGroupFamily'] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response['DBEngineVersions']
        except ClientError as err:
            logger.error(
```

```
        "Couldn't get engine versions for %s. Here's why: %s: %s", engine,
        err.response['Error']['Code'], err.response['Error']['Message'])
    raise
else:
    return versions
```

- For API details, see [DescribeDBEngineVersions in AWS SDK for Python \(Boto3\) API Reference](#).

## Describe options for DB instances

The following code example shows how to describe options for Amazon RDS DB instances.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
        instance.
        :param db_engine_version: The engine version that must be supported by the DB
        instance.
        :return: The list of DB instance options that can be used to create a
        compatible DB instance.
        """
        try:
            inst_opts = []
            paginator =
                self.rds_client.get_paginator('describe_orderable_db_instance_options')
            for page in paginator.paginate(Engine=db_engine,
                                           EngineVersion=db_engine_version):
                inst_opts += page['OrderableDBInstanceOptions']
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return inst_opts
```

- For API details, see [DescribeOrderableDBInstanceOptions](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe parameters in a DB parameter group

The following code example shows how to describe parameters in an Amazon RDS DB parameter group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:  
    """Encapsulates Amazon RDS DB instance actions."""  
    def __init__(self, rds_client):  
        """  
        :param rds_client: A Boto3 Amazon RDS client.  
        """  
        self.rds_client = rds_client  
  
    @classmethod  
    def from_client(cls):  
        """  
        Instantiates this class from a Boto3 client.  
        """  
        rds_client = boto3.client('rds')  
        return cls(rds_client)  
  
    def get_parameters(self, parameter_group_name, name_prefix='', source=None):  
        """  
        Gets the parameters that are contained in a DB parameter group.  
  
        :param parameter_group_name: The name of the parameter group to query.  
        :param name_prefix: When specified, the retrieved list of parameters is  
        filtered  
            to contain only parameters that start with this prefix.  
        :param source: When specified, only parameters from this source are retrieved.  
            For example, a source of 'user' retrieves only parameters that  
            were set by a user.  
        :return: The list of requested parameters.  
        """  
        try:  
            kwargs = {'DBParameterGroupName': parameter_group_name}  
            if source is not None:  
                kwargs['Source'] = source  
            parameters = []  
            paginator = self.rds_client.get_paginator('describe_db_parameters')  
            for page in paginator.paginate(**kwargs):  
                parameters += [  
                    p for p in page['Parameters'] if  
                    p['ParameterName'].startswith(name_prefix)]  
            except ClientError as err:  
                logger.error(  
                    "Couldn't get parameters for %s. Here's why: %s: %s",  
                    parameter_group_name,  
                    err.response['Error']['Code'], err.response['Error']['Message'])  
                raise  
            else:  
                return parameters
```

- For API details, see [DescribeDBParameters](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe snapshots of DB instances

The following code example shows how to describe snapshots of Amazon RDS DB instances.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_snapshot(self, snapshot_id):
        """
        Gets a DB instance snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_snapshots(
                DBSnapshotIdentifier=snapshot_id)
            snapshot = response['DBSnapshots'][0]
        except ClientError as err:
            logger.error(
                "Couldn't get snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
        else:
            return snapshot
```

- For API details, see [DescribeDBSnapshots](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update parameters in a DB parameter group

The following code example shows how to update parameters in an Amazon RDS DB parameter group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
```

```
"""
:param rds_client: A Boto3 Amazon RDS client.
"""
self.rds_client = rds_client

@classmethod
def from_client(cls):
    """
    Instantiates this class from a Boto3 client.
    """
    rds_client = boto3.client('rds')
    return cls(rds_client)

def update_parameters(self, parameter_group_name, update_parameters):
    """
    Updates parameters in a custom DB parameter group.

    :param parameter_group_name: The name of the parameter group to update.
    :param update_parameters: The parameters to update in the group.
    :return: Data about the modified parameter group.
    """
    try:
        response = self.rds_client.modify_db_parameter_group(
            DBParameterGroupName=parameter_group_name,
            Parameters=update_parameters)
    except ClientError as err:
        logger.error(
            "Couldn't update parameters in %s. Here's why: %s: %s",
            parameter_group_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response
```

- For API details, see [ModifyDBParameterGroup in AWS SDK for Python \(Boto3\) API Reference](#).

## Scenarios

### Get started with DB instances

The following code example shows how to:

- Create a custom DB parameter group and set parameter values.
- Create a DB instance that's configured to use the parameter group. The DB instance also contains a database.
- Take a snapshot of the instance.
- Delete the instance and parameter group.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run an interactive scenario at a command prompt.

```
class RdsInstanceScenario:
    """Runs a scenario that shows how to get started using Amazon RDS DB instances."""
    def __init__(self, instance_wrapper):
```

```

"""
:param instance_wrapper: An object that wraps Amazon RDS DB instance actions.
"""
self.instance_wrapper = instance_wrapper

def create_parameter_group(self, parameter_group_name, db_engine):
    """
    Shows how to get available engine versions for a specified database engine and
    create a DB parameter group that is compatible with a selected engine family.

    :param parameter_group_name: The name given to the newly created parameter
        group.
    :param db_engine: The database engine to use as a basis.
    :return: The newly created parameter group.
    """
    print(f"Checking for an existing DB instance parameter group named
{parameter_group_name}.")
    parameter_group =
self.instance_wrapper.get_parameter_group(parameter_group_name)
    if parameter_group is None:
        print(f"Getting available database engine versions for {db_engine}.")
        engine_versions = self.instance_wrapper.get_engine_versions(db_engine)
        families = list({ver['DBParameterGroupFamily'] for ver in engine_versions})
        family_index = q.choose("Which family do you want to use? ", families)
        print(f"Creating a parameter group.")
        self.instance_wrapper.create_parameter_group(
            parameter_group_name, families[family_index], 'Example parameter
group.')
        parameter_group =
self.instance_wrapper.get_parameter_group(parameter_group_name)
        print(f"Parameter group {parameter_group['DBParameterGroupName']}:")
        pp(parameter_group)
        print('*'*88)
    return parameter_group

def update_parameters(self, parameter_group_name):
    """
    Shows how to get the parameters contained in a custom parameter group and
    update some of the parameter values in the group.

    :param parameter_group_name: The name of the parameter group to query and
        modify.
    """
    print("Let's set some parameter values in your parameter group.")
    auto_inc_parameters = self.instance_wrapper.get_parameters(
        parameter_group_name, name_prefix='auto_increment')
    update_params = []
    for auto_inc in auto_inc_parameters:
        if auto_inc['IsModifiable'] and auto_inc['DataType'] == 'integer':
            print(f"The {auto_inc['ParameterName']} parameter is described as:")
            print(f"\t{auto_inc['Description']}")
            param_range = auto_inc['AllowedValues'].split('-')
            auto_inc['ParameterValue'] = str(q.ask(
                f"Enter a value between {param_range[0]} and {param_range[1]}: ",
                q.is_int, q.in_range(int(param_range[0]), int(param_range[1]))))
            update_params.append(auto_inc)
    self.instance_wrapper.update_parameters(parameter_group_name, update_params)
    print("You can get a list of parameters you've set by specifying a source of
'user'.")
    user_parameters = self.instance_wrapper.get_parameters(parameter_group_name,
        source='user')
    pp(user_parameters)
    print('*'*88)

def create_instance(self, instance_name, db_name, db_engine, parameter_group):
    """

```

Shows how to create a DB instance that contains a database of a specified type and is configured to use a custom DB parameter group.

```

:param instance_name: The name given to the newly created DB instance.
:param db_name: The name given to the created database.
:param db_engine: The engine of the created database.
:param parameter_group: The parameter group that is associated with the DB
instance.
:return: The newly created DB instance.
"""
print("Checking for an existing DB instance.")
db_inst = self.instance_wrapper.get_db_instance(instance_name)
if db_inst is None:
    print("Let's create a DB instance.")
    admin_username = q.ask("Enter an administrator user name for the database:",
", q.non_empty)
    admin_password = q.ask(
        "Enter a password for the administrator (at least 8 characters): ",
q.non_empty)
    engine_versions = self.instance_wrapper.get_engine_versions(
        db_engine, parameter_group['DBParameterGroupFamily'])
    engine_choices = [ver['EngineVersion'] for ver in engine_versions]
    print("The available engines for your parameter group are:")
    engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
    engine_selection = engine_versions[engine_index]
    print("The available micro DB instance classes for your database engine
are:")
    inst_opts = self.instance_wrapper.get_orderable_instances(
        engine_selection['Engine'], engine_selection['EngineVersion'])
    inst_choices = list({opt['DBInstanceClass'] for opt in inst_opts if 'micro'
in opt['DBInstanceClass']})
    inst_index = q.choose("Which micro DB instance class do you want to use? ",
inst_choices)
    group_name = parameter_group['DBParameterGroupName']
    storage_type = 'standard'
    allocated_storage = 5
    print(f"Creating a DB instance named {instance_name} and database
{db_name}.\n"
        f"The DB instance is configured to use your custom parameter group
{group_name},\n"
        f"selected engine {engine_selection['EngineVersion']}.\n"
        f"selected DB instance class {inst_choices[inst_index]},""
        f"and {allocated_storage} GiB of {storage_type} storage.\n"
        f"This typically takes several minutes.")
    db_inst = self.instance_wrapper.create_db_instance(
        db_name, instance_name, group_name, engine_selection['Engine'],
        engine_selection['EngineVersion'], inst_choices[inst_index],
storage_type,
        allocated_storage, admin_username, admin_password)
    while db_inst.get('DBInstanceState') != 'available':
        wait(10)
        db_inst = self.instance_wrapper.get_db_instance(instance_name)
print("Instance data:")
pp(db_inst)
print('-'*88)
return db_inst

@staticmethod
def display_connection(db_inst):
"""
Displays connection information about a DB instance and tips on how to
connect to it.

:param db_inst: The DB instance to display.
"""

```

```

        print("You can now connect to your database using your favorite MySql client.
\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
instance\n"
            "that is running in the same VPC as your DB instance. Pass the endpoint,
\n"
            "port, and administrator user name to 'mysql' and enter your password\n"
            "when prompted:\n")
print(f"\n\tmysql -h {db_inst['Endpoint']['Address']} -P {db_inst['Endpoint']['Port']} "
        f"-u {db_inst['MasterUsername']} -p\n")
print("For more information, see the User Guide for Amazon RDS:\n"
      "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/"
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL")
print('-'*88)

def create_snapshot(self, instance_name):
    """
    Shows how to create a DB instance snapshot and wait until it's available.

    :param instance_name: The name of a DB instance to snapshot.
    """
    if q.ask("Do you want to create a snapshot of your DB instance (y/n)? ",
q.is_yesno):
        snapshot_id = f"{instance_name}-{uuid.uuid4()}"
        print(f"Creating a snapshot named {snapshot_id}. This typically takes a few
minutes.")
        snapshot = self.instance_wrapper.create_snapshot(snapshot_id,
instance_name)
        while snapshot.get('Status') != 'available':
            wait(10)
            snapshot = self.instance_wrapper.get_snapshot(snapshot_id)
pp(snapshot)
print('-'*88)

def cleanup(self, db_inst, parameter_group_name):
    """
    Shows how to clean up a DB instance and parameter group.
    Before the parameter group can be deleted, all associated DB instances must
first
    be deleted.

    :param db_inst: The DB instance to delete.
    :param parameter_group_name: The DB parameter group to delete.
    """
    if q.ask(
            "\nDo you want to delete the DB instance and parameter group (y/n)? ",
            q.is_yesno):
        print(f"Deleting DB instance {db_inst['DBInstanceIdentifier']}.")
        self.instance_wrapper.delete_db_instance(db_inst['DBInstanceIdentifier'])
        print("Waiting for the DB instance to delete. This typically takes several
minutes.")
        while db_inst is not None:
            wait(10)
            db_inst =
self.instance_wrapper.get_db_instance(db_inst['DBInstanceIdentifier'])
            print(f"Deleting parameter group {parameter_group_name}.")
            self.instance_wrapper.delete_parameter_group(parameter_group_name)

def run_scenario(
        self, db_engine, parameter_group_name, instance_name, db_name):
logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

print('-'*88)
print("Welcome to the Amazon Relational Database Service (Amazon RDS)\n"
      "get started with DB instances demo.")

```

```

print('*'*88)

parameter_group = self.create_parameter_group(parameter_group_name, db_engine)
self.update_parameters(parameter_group_name)
db_inst = self.create_instance(instance_name, db_name, db_engine,
parameter_group)
self.display_connection(db_inst)
self.create_snapshot(instance_name)
self.cleanup(db_inst, parameter_group_name)

print("\nThanks for watching!")
print('*'*88)

if __name__ == '__main__':
    try:
        scenario = RdsInstanceScenario(InstanceWrapper.from_client())
        scenario.run_scenario(
            'mysql', 'doc-example-parameter-group', 'doc-example-instance',
            'doceexampledb')
    except Exception:
        logging.exception("Something went wrong with the demo.")

```

Define functions that are called by the scenario to manage Amazon RDS actions.

```

class InstanceWrapper:
    """Encapsulates Amazon RDS DB instance actions."""
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon RDS client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client('rds')
        return cls(rds_client)

    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The parameter group.
        """
        try:
            response = self.rds_client.describe_db_parameter_groups(
                DBParameterGroupName=parameter_group_name)
            parameter_group = response['DBParameterGroups'][0]
        except ClientError as err:
            if err.response['Error']['Code'] == 'DBParameterGroupNotFound':
                logger.info("Parameter group %s does not exist.", parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response['Error']['Code'], err.response['Error']['Message'])
                raise
        else:
            return parameter_group

```

```
def create_parameter_group(self, parameter_group_name, parameter_group_family,
                           description):
    """
    Creates a DB parameter group that is based on the specified parameter group
    family.

    :param parameter_group_name: The name of the newly created parameter group.
    :param parameter_group_family: The family that is used as the basis of the new
                                   parameter group.
    :param description: A description given to the parameter group.
    :return: Data about the newly created parameter group.
    """
    try:
        response = self.rds_client.create_db_parameter_group(
            DBParameterGroupName=parameter_group_name,
            DBParameterGroupFamily=parameter_group_family,
            Description=description)
    except ClientError as err:
        logger.error(
            "Couldn't create parameter group %s. Here's why: %s: %s",
            parameter_group_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
    else:
        return response

def delete_parameter_group(self, parameter_group_name):
    """
    Deletes a DB parameter group.

    :param parameter_group_name: The name of the parameter group to delete.
    :return: Data about the parameter group.
    """
    try:
        self.rds_client.delete_db_parameter_group(
            DBParameterGroupName=parameter_group_name)
    except ClientError as err:
        logger.error(
            "Couldn't delete parameter group %s. Here's why: %s: %s",
            parameter_group_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise

def get_parameters(self, parameter_group_name, name_prefix='', source=None):
    """
    Gets the parameters that are contained in a DB parameter group.

    :param parameter_group_name: The name of the parameter group to query.
    :param name_prefix: When specified, the retrieved list of parameters is
                       filtered
                           to contain only parameters that start with this prefix.
    :param source: When specified, only parameters from this source are retrieved.
                  For example, a source of 'user' retrieves only parameters that
                  were set by a user.
    :return: The list of requested parameters.
    """
    try:
        kwargs = {'DBParameterGroupName': parameter_group_name}
        if source is not None:
            kwargs['Source'] = source
        parameters = []
        paginator = self.rds_client.getPaginator('describe_db_parameters')
        for page in paginator.paginate(**kwargs):
            parameters += [
                p for p in page['Parameters'] if
                p['ParameterName'].startswith(name_prefix)]
    except ClientError as err:
        logger.error(
            "Couldn't get parameters. Here's why: %s: %s",
            parameter_group_name,
            err.response['Error']['Code'], err.response['Error']['Message'])
        raise
```

```
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return parameters

    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_parameter_group(
                DBParameterGroupName=parameter_group_name,
                Parameters=update_parameters)
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response

    def create_snapshot(self, snapshot_id, instance_id):
        """
        Creates a snapshot of a DB instance.

        :param snapshot_id: The ID to give the created snapshot.
        :param instance_id: The ID of the DB instance to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_snapshot(
                DBSnapshotIdentifier=snapshot_id, DBInstanceIdentifier=instance_id)
            snapshot = response['DBSnapshot']
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                instance_id, err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return snapshot

    def get_snapshot(self, snapshot_id):
        """
        Gets a DB instance snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_snapshots(
                DBSnapshotIdentifier=snapshot_id)
            snapshot = response['DBSnapshots'][0]
        except ClientError as err:
            logger.error(
                "Couldn't get snapshot %s. Here's why: %s: %s",
                snapshot_id, err.response['Error']['Code'], err.response['Error']['Message'])
            raise
```

```

        else:
            return snapshot

    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned list of
                                       engine versions to those that are compatible
                                       with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {'Engine': engine}
            if parameter_group_family is not None:
                kwargs['DBParameterGroupFamily'] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response['DBEngineVersions']
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
        else:
            return versions

    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
                          instance.
        :param db_engine_version: The engine version that must be supported by the DB
                                  instance.
        :return: The list of DB instance options that can be used to create a
                 compatible DB instance.
        """
        try:
            inst_opts = []
            paginator =
self.rds_client.get_paginator('describe_orderable_db_instance_options')
            for page in paginator.paginate(Engine=db_engine,
                                         EngineVersion=db_engine_version):
                inst_opts += page['OrderableDBInstanceOptions']
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
        else:
            return inst_opts

    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id)

```

```

        db_inst = response['DBInstances'][0]
    except ClientError as err:
        if err.response['Error']['Code'] == 'DBInstanceNotFound':
            logger.info("Instance %s does not exist.", instance_id)
        else:
            logger.error(
                "Couldn't get DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
    else:
        return db_inst

    def create_db_instance(
        self, db_name, instance_id, parameter_group_name, db_engine,
        db_engine_version,
        instance_class, storage_type, allocated_storage, admin_name,
        admin_password):
        """
        Creates a DB instance.

        :param db_name: The name of the database that is created in the DB instance.
        :param instance_id: The ID to give the newly created DB instance.
        :param parameter_group_name: A parameter group to associate with the DB
        instance.
        :param db_engine: The database engine of a database to create in the DB
        instance.
        :param db_engine_version: The engine version for the created database.
        :param instance_class: The DB instance class for the newly created DB instance.
        :param storage_type: The storage type of the DB instance.
        :param allocated_storage: The amount of storage allocated on the DB instance,
        in GiBs.
        :param admin_name: The name of the admin user for the created database.
        :param admin_password: The admin password for the created database.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBName=db_name,
                DBInstanceIdentifier=instance_id,
                DBParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                DBInstanceClass=instance_class,
                StorageType=storage_type,
                AllocatedStorage=allocated_storage,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password)
            db_inst = response['DBInstance']
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response['Error']['Code'],
                err.response['Error']['Message'])
            raise
        else:
            return db_inst

    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id, SkipFinalSnapshot=True,

```

```
        DeleteAutomatedBackups=True)
    db_inst = response['DBInstance']
except ClientError as err:
    logger.error(
        "Couldn't delete DB instance %s. Here's why: %s: %s",
        instance_id,
        err.response['Error']['Code'],
        err.response['Error']['Message'])
    raise
else:
    return db_inst
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateDBInstance](#)
  - [CreateDBParameterGroup](#)
  - [CreateDBSnapshot](#)
  - [DeleteDBInstance](#)
  - [DeleteDBParameterGroup](#)
  - [DescribeDBEngineVersions](#)
  - [DescribeDBInstances](#)
  - [DescribeDBParameterGroups](#)
  - [DescribeDBParameters](#)
  - [DescribeDBSchemas](#)
  - [DescribeOrderableDBInstanceOptions](#)
  - [ModifyDBParameterGroup](#)

## Amazon Rekognition examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Rekognition.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4132\)](#)
- [Scenarios \(p. 4147\)](#)

## Actions

### Compare faces in an image against a reference image

The following code example shows how to compare faces in an image against a reference image with Amazon Rekognition.

For more information, see [Comparing faces in images](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionImage:  
    """  
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper  
    around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, image, image_name, rekognition_client):  
        """  
        Initializes the image object.  
  
        :param image: Data that defines the image, either the image bytes or  
                     an Amazon S3 bucket and object key.  
        :param image_name: The name of the image.  
        :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.image = image  
        self.image_name = image_name  
        self.rekognition_client = rekognition_client  
  
    def compare_faces(self, target_image, similarity):  
        """  
        Compares faces in the image with the largest face in the target image.  
  
        :param target_image: The target image to compare against.  
        :param similarity: Faces in the image must have a similarity value greater  
                           than this value to be included in the results.  
        :return: A tuple. The first element is the list of faces that match the  
                reference image. The second element is the list of faces that have  
                a similarity value below the specified threshold.  
        """  
        try:  
            response = self.rekognition_client.compare_faces(  
                SourceImage=self.image,  
                TargetImage=target_image.image,  
                SimilarityThreshold=similarity)  
            matches = [RekognitionFace(match['Face']) for match  
                      in response['FaceMatches']]  
            unmatches = [RekognitionFace(face) for face in response['UnmatchedFaces']]  
            logger.info(  
                "Found %s matched faces and %s unmatched faces.",  
                len(matches), len(unmatches))  
        except ClientError:  
            logger.exception(  
                "Couldn't match faces from %s to %s.", self.image_name,  
                target_image.image_name)  
            raise  
        else:  
            return matches, unmatches
```

- For API details, see [CompareFaces in AWS SDK for Python \(Boto3\) API Reference](#).

## Create a collection

The following code example shows how to create an Amazon Rekognition collection.

For more information, see [Creating a collection](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollectionManager:  
    """  
        Encapsulates Amazon Rekognition collection management functions.  
        This class is a thin wrapper around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, rekognition_client):  
        """  
            Initializes the collection manager object.  
            :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.rekognition_client = rekognition_client  
  
    def create_collection(self, collection_id):  
        """  
            Creates an empty collection.  
            :param collection_id: Text that identifies the collection.  
            :return: The newly created collection.  
        """  
        try:  
            response = self.rekognition_client.create_collection(  
                CollectionId=collection_id)  
            response['CollectionId'] = collection_id  
            collection = RekognitionCollection(response, self.rekognition_client)  
            logger.info("Created collection %s.", collection_id)  
        except ClientError:  
            logger.exception("Couldn't create collection %s.", collection_id)  
            raise  
        else:  
            return collection
```

- For API details, see [CreateCollection in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete a collection

The following code example shows how to delete an Amazon Rekognition collection.

For more information, see [Deleting a collection](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollection:  
    """  
        Encapsulates an Amazon Rekognition collection. This class is a thin wrapper  
        around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, collection, rekognition_client):  
        """  
            Initializes a collection object.  
            :param collection: Collection data in the format returned by a call to  
                create_collection.  
            :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.collection_id = collection['CollectionId']
```

```
        self.collection_arn, self.face_count, self.created = self._unpack_collection(
            collection)
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get('CollectionArn'),
            collection.get('FaceCount', 0),
            collection.get('CreationTimestamp'))

    def delete_collection(self):
        """
        Deletes the collection.
        """
        try:
            self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
            logger.exception("Couldn't delete collection %s.", self.collection_id)
            raise
```

- For API details, see [DeleteCollection in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete faces from a collection

The following code example shows how to delete faces from an Amazon Rekognition collection.

For more information, see [Deleting faces from a collection](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection['CollectionId']
        self.collection_arn, self.face_count, self.created = self._unpack_collection(
            collection)
        self.rekognition_client = rekognition_client
```

```
@staticmethod
def _unpack_collection(collection):
    """
    Unpacks optional parts of a collection that can be returned by
    describe_collection.

    :param collection: The collection data.
    :return: A tuple of the data in the collection.
    """
    return (
        collection.get('CollectionArn'),
        collection.get('FaceCount', 0),
        collection.get('CreationTimestamp'))

def delete_faces(self, face_ids):
    """
    Deletes faces from the collection.

    :param face_ids: The list of IDs of faces to delete.
    :return: The list of IDs of faces that were deleted.
    """
    try:
        response = self.rekognition_client.delete_faces(
            CollectionId=self.collection_id, FaceIds=face_ids)
        deleted_ids = response['DeletedFaces']
        logger.info(
            "Deleted %s faces from %s.", len(deleted_ids), self.collection_id)
    except ClientError:
        logger.exception("Couldn't delete faces from %s.", self.collection_id)
        raise
    else:
        return deleted_ids
```

- For API details, see [DeleteFaces in AWS SDK for Python \(Boto3\) API Reference](#).

## Describe a collection

The following code example shows how to describe an Amazon Rekognition collection.

For more information, see [Describing a collection](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """

```

```
    self.collection_id = collection['CollectionId']
    self.collection_arn, self.face_count, self.created = self._unpack_collection(
        collection)
    self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get('CollectionArn'),
            collection.get('FaceCount', 0),
            collection.get('CreationTimestamp'))

    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        """
        try:
            response = self.rekognition_client.describe_collection(
                CollectionId=self.collection_id)
            # Work around capitalization of Arn vs. ARN
            response['CollectionArn'] = response.get('CollectionARN')
            (self.collection_arn, self.face_count,
             self.created) = self._unpack_collection(response)
            logger.info("Got data for collection %s.", self.collection_id)
        except ClientError:
            logger.exception("Couldn't get data for collection %s.",
                             self.collection_id)
            raise
        else:
            return self.to_dict()
```

- For API details, see [DescribeCollection](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect faces in an image

The following code example shows how to detect faces in an image with Amazon Rekognition.

For more information, see [Detecting faces in an image](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, image, image_name, rekognition_client):
```

```
Initializes the image object.

:param image: Data that defines the image, either the image bytes or
              an Amazon S3 bucket and object key.
:param image_name: The name of the image.
:param rekognition_client: A Boto3 Rekognition client.
"""
self.image = image
self.image_name = image_name
self.rekognition_client = rekognition_client

def detect_faces(self):
    """
    Detects faces in the image.

    :return: The list of faces found in the image.
    """
try:
    response = self.rekognition_client.detect_faces(
        Image=self.image, Attributes=['ALL'])
    faces = [RekognitionFace(face) for face in response['FaceDetails']]
    logger.info("Detected %s faces.", len(faces))
except ClientError:
    logger.exception("Couldn't detect faces in %s.", self.image_name)
    raise
else:
    return faces
```

- For API details, see [DetectFaces in AWS SDK for Python \(Boto3\) API Reference](#).

## Detect labels in an image

The following code example shows how to detect labels in an image with Amazon Rekognition.

For more information, see [Detecting labels in an image](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client

    def detect_labels(self, max_labels):
```

```
"""
    Detects labels in the image. Labels are objects and people.

:param max_labels: The maximum number of labels to return.
:return: The list of labels detected in the image.
"""
try:
    response = self.rekognition_client.detect_labels(
        Image=self.image, MaxLabels=max_labels)
    labels = [RekognitionLabel(label) for label in response['Labels']]
    logger.info("Found %s labels in %s.", len(labels), self.image_name)
except ClientError:
    logger.info("Couldn't detect labels in %s.", self.image_name)
    raise
else:
    return labels
```

- For API details, see [DetectLabels](#) in *AWS SDK for Python (Boto3) API Reference*.

### Detect moderation labels in an image

The following code example shows how to detect moderation labels in an image with Amazon Rekognition. Moderation labels identify content that may be inappropriate for some audiences.

For more information, see [Detecting inappropriate images](#).

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionImage:
    """
        Encapsulates an Amazon Rekognition image. This class is a thin wrapper
        around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, image, image_name, rekognition_client):
        """
            Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client

    def detect_moderation_labels(self):
        """
            Detects moderation labels in the image. Moderation labels identify content
            that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image)
            labels = [RekognitionModerationLabel(label)]
        except ClientError:
            logger.info("Couldn't detect moderation labels in %s.", self.image_name)
            raise
        else:
            return labels
```

```
        for label in response['ModerationLabels']]
    logger.info(
        "Found %s moderation labels in %s.", len(labels), self.image_name)
except ClientError:
    logger.exception(
        "Couldn't detect moderation labels in %s.", self.image_name)
    raise
else:
    return labels
```

- For API details, see [DetectModerationLabels](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect text in an image

The following code example shows how to detect text in an image with Amazon Rekognition.

For more information, see [Detecting text in an image](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client

    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in response['TextDetections']]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

- For API details, see [DetectText](#) in *AWS SDK for Python (Boto3) API Reference*.

## Index faces to a collection

The following code example shows how to index faces in an image and add them to an Amazon Rekognition collection.

For more information, see [Adding faces to a collection](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollection:  
    """  
        Encapsulates an Amazon Rekognition collection. This class is a thin wrapper  
        around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, collection, rekognition_client):  
        """  
            Initializes a collection object.  
  
        :param collection: Collection data in the format returned by a call to  
                          create_collection.  
        :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.collection_id = collection['CollectionId']  
        self.collection_arn, self.face_count, self.created = self._unpack_collection(  
            collection)  
        self.rekognition_client = rekognition_client  
  
    @staticmethod  
    def _unpack_collection(collection):  
        """  
            Unpacks optional parts of a collection that can be returned by  
            describe_collection.  
  
        :param collection: The collection data.  
        :return: A tuple of the data in the collection.  
        """  
        return (  
            collection.get('CollectionArn'),  
            collection.get('FaceCount', 0),  
            collection.get('CreationTimestamp'))  
  
    def index_faces(self, image, max_faces):  
        """  
            Finds faces in the specified image, indexes them, and stores them in the  
            collection.  
  
        :param image: The image to index.  
        :param max_faces: The maximum number of faces to index.  
        :return: A tuple. The first element is a list of indexed faces.  
                The second element is a list of faces that couldn't be indexed.  
        """  
        try:  
            response = self.rekognition_client.index_faces(  
                CollectionId=self.collection_id, Image=image.image,  
                ExternalImageId=image.image_name, MaxFaces=max_faces,  
                DetectionAttributes=['ALL'])  
            indexed_faces = [  
                RekognitionFace({**face['Face'], **face['FaceDetail']})  
                for face in response['FaceRecords']]  
            unindexed_faces = [  
        
```

```
        RekognitionFace(face['FaceDetail'])
        for face in response['UnindexedFaces']]
    logger.info(
        "Indexed %s faces in %s. Could not index %s faces.",
        len(indexed_faces),
        image.image_name, len(unindexed_faces))
except ClientError:
    logger.exception("Couldn't index faces in image %s.", image.image_name)
    raise
else:
    return indexed_faces, unindexed_faces
```

- For API details, see [IndexFaces](#) in *AWS SDK for Python (Boto3) API Reference*.

## List collections

The following code example shows how to list Amazon Rekognition collections.

For more information, see [Listing collections](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client

    def list_collections(self, max_results):
        """
        Lists collections for the current account.

        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
            response = self.rekognition_client.list_collections(MaxResults=max_results)
            collections = [
                RekognitionCollection({'CollectionId': col_id},
            self.rekognition_client)
                    for col_id in response['CollectionIds']]
        except ClientError:
            logger.exception("Couldn't list collections.")
            raise
        else:
            return collections
```

- For API details, see [ListCollections](#) in *AWS SDK for Python (Boto3) API Reference*.

## List faces in a collection

The following code example shows how to list faces in an Amazon Rekognition collection.

For more information, see [Listing faces in a collection](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollection:  
    """  
        Encapsulates an Amazon Rekognition collection. This class is a thin wrapper  
        around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, collection, rekognition_client):  
        """  
            Initializes a collection object.  
  
        :param collection: Collection data in the format returned by a call to  
                          create_collection.  
        :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.collection_id = collection['CollectionId']  
        self.collection_arn, self.face_count, self.created = self._unpack_collection(  
            collection)  
        self.rekognition_client = rekognition_client  
  
    @staticmethod  
    def _unpack_collection(collection):  
        """  
            Unpacks optional parts of a collection that can be returned by  
            describe_collection.  
  
        :param collection: The collection data.  
        :return: A tuple of the data in the collection.  
        """  
        return (  
            collection.get('CollectionArn'),  
            collection.get('FaceCount', 0),  
            collection.get('CreationTimestamp'))  
  
    def list_faces(self, max_results):  
        """  
            Lists the faces currently indexed in the collection.  
  
        :param max_results: The maximum number of faces to return.  
        :return: The list of faces in the collection.  
        """  
        try:  
            response = self.rekognition_client.list_faces(  
                CollectionId=self.collection_id, MaxResults=max_results)  
            faces = [RekognitionFace(face) for face in response['Faces']]  
            logger.info(  
                "Found %s faces in collection %s.", len(faces), self.collection_id)  
        except ClientError:  
            logger.exception(  
                "Couldn't list faces in collection %s.", self.collection_id)  
            raise  
        else:  
            return faces
```

- For API details, see [ListFaces](#) in *AWS SDK for Python (Boto3) API Reference*.

## Recognize celebrities in an image

The following code example shows how to recognize celebrities in an image with Amazon Rekognition.

For more information, see [Recognizing celebrities in an image](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionImage:  
    """  
        Encapsulates an Amazon Rekognition image. This class is a thin wrapper  
        around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, image, image_name, rekognition_client):  
        """  
            Initializes the image object.  
  
            :param image: Data that defines the image, either the image bytes or  
                         an Amazon S3 bucket and object key.  
            :param image_name: The name of the image.  
            :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.image = image  
        self.image_name = image_name  
        self.rekognition_client = rekognition_client  
  
    def recognize_celebrities(self):  
        """  
            Detects celebrities in the image.  
  
            :return: A tuple. The first element is the list of celebrities found in  
                    the image. The second element is the list of faces that were  
                    detected but did not match any known celebrities.  
        """  
        try:  
            response = self.rekognition_client.recognize_celebrities(  
                Image=self.image)  
            celebrities = [RekognitionCelebrity(celeb)  
                           for celeb in response['CelebrityFaces']]  
            other_faces = [RekognitionFace(face)  
                           for face in response['UnrecognizedFaces']]  
            logger.info(  
                "Found %s celebrities and %s other faces in %s.", len(celebrities),  
                len(other_faces), self.image_name)  
        except ClientError:  
            logger.exception("Couldn't detect celebrities in %s.", self.image_name)  
            raise  
        else:  
            return celebrities, other_faces
```

- For API details, see [RecognizeCelebrities](#) in *AWS SDK for Python (Boto3) API Reference*.

## Search for faces in a collection

The following code example shows how to search for faces in an Amazon Rekognition collection that match another face from the collection.

For more information, see [Searching for a face \(face ID\)](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollection:  
    """  
        Encapsulates an Amazon Rekognition collection. This class is a thin wrapper  
        around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, collection, rekognition_client):  
        """  
            Initializes a collection object.  
  
        :param collection: Collection data in the format returned by a call to  
                          create_collection.  
        :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.collection_id = collection['CollectionId']  
        self.collection_arn, self.face_count, self.created = self._unpack_collection(  
            collection)  
        self.rekognition_client = rekognition_client  
  
    @staticmethod  
    def _unpack_collection(collection):  
        """  
            Unpacks optional parts of a collection that can be returned by  
            describe_collection.  
  
        :param collection: The collection data.  
        :return: A tuple of the data in the collection.  
        """  
        return (  
            collection.get('CollectionArn'),  
            collection.get('FaceCount', 0),  
            collection.get('CreationTimestamp'))  
  
    def search_faces(self, face_id, threshold, max_faces):  
        """  
            Searches for faces in the collection that match another face from the  
            collection.  
  
        :param face_id: The ID of the face in the collection to search for.  
        :param threshold: The match confidence must be greater than this value  
                          for a face to be included in the results.  
        :param max_faces: The maximum number of faces to return.  
        :return: The list of matching faces found in the collection. This list does  
                not contain the face specified by `face_id`.  
        """  
        try:  
            response = self.rekognition_client.search_faces(  
                CollectionId=self.collection_id, FaceId=face_id,  
                FaceMatchThreshold=threshold, MaxFaces=max_faces)  
            faces = [RekognitionFace(face['Face']) for face in response['FaceMatches']]  
            logger.info(  
                "Found %s faces in %s that match %s.", len(faces), self.collection_id,
```

```
        face_id)
    except ClientError:
        logger.exception(
            "Couldn't search for faces in %s that match %s.", self.collection_id,
            face_id)
        raise
    else:
        return faces
```

- For API details, see [SearchFaces](#) in *AWS SDK for Python (Boto3) API Reference*.

### Search for faces in a collection compared to a reference image

The following code example shows how to search for faces in an Amazon Rekognition collection compared to a reference image.

For more information, see [Searching for a face \(image\)](#).

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection['CollectionId']
        self.collection_arn, self.face_count, self.created = self._unpack_collection(
            collection)
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get('CollectionArn'),
            collection.get('FaceCount', 0),
            collection.get('CreationTimestamp'))

    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.
```

```
:param image: The image that contains the reference face to search for.
:param threshold: The match confidence must be greater than this value
                  for a face to be included in the results.
:param max_faces: The maximum number of faces to return.
:return: A tuple. The first element is the face found in the reference image.
         The second element is the list of matching faces found in the
         collection.
"""
try:
    response = self.rekognition_client.search_faces_by_image(
        CollectionId=self.collection_id, Image=image.image,
        FaceMatchThreshold=threshold, MaxFaces=max_faces)
    image_face = RekognitionFace({
        'BoundingBox': response['SearchedFaceBoundingBox'],
        'Confidence': response['SearchedFaceConfidence']
    })
    collection_faces = [
        RekognitionFace(face['Face']) for face in response['FaceMatches']]
    logger.info("Found %s faces in the collection that match the largest "
                "face in %s.", len(collection_faces), image.image_name)
except ClientError:
    logger.exception(
        "Couldn't search for faces in %s that match %s.", self.collection_id,
        image.image_name)
    raise
else:
    return image_face, collection_faces
```

- For API details, see [SearchFacesByImage](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Build a collection and find faces in it

The following code example shows how to:

- Create an Amazon Rekognition collection.
- Add images to the collection and detect faces in it.
- Search the collection for faces that match a reference image.
- Delete a collection.

For more information, see [Searching faces in a collection](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create classes that wrap Amazon Rekognition functions.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
from rekognition_objects import RekognitionFace
from rekognition_image_detection import RekognitionImage

logger = logging.getLogger(__name__)
```

```
class RekognitionImage:  
    """  
        Encapsulates an Amazon Rekognition image. This class is a thin wrapper  
        around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, image, image_name, rekognition_client):  
        """  
            Initializes the image object.  
  
            :param image: Data that defines the image, either the image bytes or  
                         an Amazon S3 bucket and object key.  
            :param image_name: The name of the image.  
            :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.image = image  
        self.image_name = image_name  
        self.rekognition_client = rekognition_client  
  
    @classmethod  
    def from_file(cls, image_file_name, rekognition_client, image_name=None):  
        """  
            Creates a RekognitionImage object from a local file.  
  
            :param image_file_name: The file name of the image. The file is opened and its  
                                  bytes are read.  
            :param rekognition_client: A Boto3 Rekognition client.  
            :param image_name: The name of the image. If this is not specified, the  
                              file name is used as the image name.  
            :return: The RekognitionImage object, initialized with image bytes from the  
                    file.  
        """  
        with open(image_file_name, 'rb') as img_file:  
            image = {'Bytes': img_file.read()}  
        name = image_file_name if image_name is None else image_name  
        return cls(image, name, rekognition_client)  
  
class RekognitionCollectionManager:  
    """  
        Encapsulates Amazon Rekognition collection management functions.  
        This class is a thin wrapper around parts of the Boto3 Amazon Rekognition API.  
    """  
    def __init__(self, rekognition_client):  
        """  
            Initializes the collection manager object.  
  
            :param rekognition_client: A Boto3 Rekognition client.  
        """  
        self.rekognition_client = rekognition_client  
  
    def create_collection(self, collection_id):  
        """  
            Creates an empty collection.  
  
            :param collection_id: Text that identifies the collection.  
            :return: The newly created collection.  
        """  
        try:  
            response = self.rekognition_client.create_collection(  
                CollectionId=collection_id)  
            response['CollectionId'] = collection_id  
            collection = RekognitionCollection(response, self.rekognition_client)  
            logger.info("Created collection %s.", collection_id)  
        except ClientError:  
            logger.exception("Couldn't create collection %s.", collection_id)
```

```
        raise
    else:
        return collection

def list_collections(self, max_results):
    """
    Lists collections for the current account.

    :param max_results: The maximum number of collections to return.
    :return: The list of collections for the current account.
    """
    try:
        response = self.rekognition_client.list_collections(MaxResults=max_results)
        collections = [
            RekognitionCollection({'CollectionId': col_id},
self.rekognition_client)
            for col_id in response['CollectionIds']]
    except ClientError:
        logger.exception("Couldn't list collections.")
        raise
    else:
        return collections

class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection['CollectionId']
        self.collection_arn, self.face_count, self.created = self._unpack_collection(
            collection)
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get('CollectionArn'),
            collection.get('FaceCount', 0),
            collection.get('CreationTimestamp'))

    def to_dict(self):
        """
        Renders parts of the collection data to a dict.

        :return: The collection data as a dict.
        """
        rendering = {
            'collection_id': self.collection_id,
            'collection_arn': self.collection_arn,
            'face_count': self.face_count,
            'created': self.created}
```

```

        }

    return rendering

def describe_collection(self):
    """
    Gets data about the collection from the Amazon Rekognition service.

    :return: The collection rendered as a dict.
    """
    try:
        response = self.rekognition_client.describe_collection(
            CollectionId=self.collection_id)
        # Work around capitalization of Arn vs. ARN
        response['CollectionArn'] = response.get('CollectionARN')
        (self.collection_arn, self.face_count,
         self.created) = self._unpack_collection(response)
        logger.info("Got data for collection %s.", self.collection_id)
    except ClientError:
        logger.exception("Couldn't get data for collection %s.",
                         self.collection_id)
        raise
    else:
        return self.to_dict()

def delete_collection(self):
    """
    Deletes the collection.
    """
    try:
        self.rekognition_client.delete_collection(CollectionId=self.collection_id)
        logger.info("Deleted collection %s.", self.collection_id)
        self.collection_id = None
    except ClientError:
        logger.exception("Couldn't delete collection %s.", self.collection_id)
        raise

def index_faces(self, image, max_faces):
    """
    Finds faces in the specified image, indexes them, and stores them in the
    collection.

    :param image: The image to index.
    :param max_faces: The maximum number of faces to index.
    :return: A tuple. The first element is a list of indexed faces.
            The second element is a list of faces that couldn't be indexed.
    """
    try:
        response = self.rekognition_client.index_faces(
            CollectionId=self.collection_id, Image=image.image,
            ExternalImageId=image.image_name, MaxFaces=max_faces,
            DetectionAttributes=['ALL'])
        indexed_faces = [
            RekognitionFace({**face['Face'], **face['FaceDetail']})
            for face in response['FaceRecords']]
        unindexed_faces = [
            RekognitionFace(face['FaceDetail'])
            for face in response['UnindexedFaces']]
        logger.info(
            "Indexed %s faces in %s. Could not index %s faces.",
            len(indexed_faces),
            image.image_name, len(unindexed_faces))
    except ClientError:
        logger.exception("Couldn't index faces in image %s.", image.image_name)
        raise
    else:
        return indexed_faces, unindexed_faces

```

```
def list_faces(self, max_results):
    """
    Lists the faces currently indexed in the collection.

    :param max_results: The maximum number of faces to return.
    :return: The list of faces in the collection.
    """
    try:
        response = self.rekognition_client.list_faces(
            CollectionId=self.collection_id, MaxResults=max_results)
        faces = [RekognitionFace(face) for face in response['Faces']]
        logger.info(
            "Found %s faces in collection %s.", len(faces), self.collection_id)
    except ClientError:
        logger.exception(
            "Couldn't list faces in collection %s.", self.collection_id)
        raise
    else:
        return faces

def search_faces(self, face_id, threshold, max_faces):
    """
    Searches for faces in the collection that match another face from the
    collection.

    :param face_id: The ID of the face in the collection to search for.
    :param threshold: The match confidence must be greater than this value
                      for a face to be included in the results.
    :param max_faces: The maximum number of faces to return.
    :return: The list of matching faces found in the collection. This list does
            not contain the face specified by `face_id`.
    """
    try:
        response = self.rekognition_client.search_faces(
            CollectionId=self.collection_id, FaceId=face_id,
            FaceMatchThreshold=threshold, MaxFaces=max_faces)
        faces = [RekognitionFace(face['Face']) for face in response['FaceMatches']]
        logger.info(
            "Found %s faces in %s that match %s.", len(faces), self.collection_id,
            face_id)
    except ClientError:
        logger.exception(
            "Couldn't search for faces in %s that match %s.", self.collection_id,
            face_id)
        raise
    else:
        return faces

def search_faces_by_image(self, image, threshold, max_faces):
    """
    Searches for faces in the collection that match the largest face in the
    reference image.

    :param image: The image that contains the reference face to search for.
    :param threshold: The match confidence must be greater than this value
                      for a face to be included in the results.
    :param max_faces: The maximum number of faces to return.
    :return: A tuple. The first element is the face found in the reference image.
            The second element is the list of matching faces found in the
            collection.
    """
    try:
        response = self.rekognition_client.search_faces_by_image(
            CollectionId=self.collection_id, Image=image.image,
            FaceMatchThreshold=threshold, MaxFaces=max_faces)
```

```

        image_face = RekognitionFace({
            'BoundingBox': response['SearchedFaceBoundingBox'],
            'Confidence': response['SearchedFaceConfidence']
        })
        collection_faces = [
            RekognitionFace(face['Face']) for face in response['FaceMatches']]
        logger.info("Found %s faces in the collection that match the largest "
                    "face in %s.", len(collection_faces), image.image_name)
    except ClientError:
        logger.exception(
            "Couldn't search for faces in %s that match %s.", self.collection_id,
            image.image_name)
        raise
    else:
        return image_face, collection_faces

class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""
    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
                     functions.
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
        """
        self.bounding_box = face.get('BoundingBox')
        self.confidence = face.get('Confidence')
        self.landmarks = face.get('Landmarks')
        self.pose = face.get('Pose')
        self.quality = face.get('Quality')
        age_range = face.get('AgeRange')
        if age_range is not None:
            self.age_range = (age_range.get('Low'), age_range.get('High'))
        else:
            self.age_range = None
        self.smile = face.get('Smile', {}).get('Value')
        self.eyeglasses = face.get('Eyeglasses', {}).get('Value')
        self.sunglasses = face.get('Sunglasses', {}).get('Value')
        self.gender = face.get('Gender', {}).get('Value', None)
        self.beard = face.get('Beard', {}).get('Value')
        self.mustache = face.get('Mustache', {}).get('Value')
        self.eyes_open = face.get('EyesOpen', {}).get('Value')
        self.mouth_open = face.get('MouthOpen', {}).get('Value')
        self.emotions = [emo.get('Type') for emo in face.get('Emotions', [])
                        if emo.get('Confidence', 0) > 50]
        self.face_id = face.get('FaceId')
        self.image_id = face.get('ImageId')
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
        rendering = {}
        if self.bounding_box is not None:
            rendering['bounding_box'] = self.bounding_box
        if self.age_range is not None:
            rendering['age'] = f'{self.age_range[0]} - {self.age_range[1]}'
        if self.gender is not None:
            rendering['gender'] = self.gender
        if self.emotions:
            rendering['emotions'] = self.emotions

```

```

if self.face_id is not None:
    rendering['face_id'] = self.face_id
if self.image_id is not None:
    rendering['image_id'] = self.image_id
if self.timestamp is not None:
    rendering['timestamp'] = self.timestamp
has = []
if self.smile:
    has.append('smile')
if self.eyeglasses:
    has.append('eyeglasses')
if self.sunglasses:
    has.append('sunglasses')
if self.beard:
    has.append('beard')
if self.mustache:
    has.append('mustache')
if self.eyes_open:
    has.append('open eyes')
if self.mouth_open:
    has.append('open mouth')
if has:
    rendering['has'] = has
return rendering

```

Use the wrapper classes to build a collection of faces from a set of images and then search for faces in the collection.

```

def usage_demo():
    print('*'*88)
    print("Welcome to the Amazon Rekognition face collection demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    rekognition_client = boto3.client('rekognition')
    images = [
        RekognitionImage.from_file(
            '.media/pexels-agung-pandit-wiguna-1128316.jpg', rekognition_client,
            image_name='sitting'),
        RekognitionImage.from_file(
            '.media/pexels-agung-pandit-wiguna-1128317.jpg', rekognition_client,
            image_name='hopping'),
        RekognitionImage.from_file(
            '.media/pexels-agung-pandit-wiguna-1128318.jpg', rekognition_client,
            image_name='biking')]

    collection_mgr = RekognitionCollectionManager(rekognition_client)
    collection = collection_mgr.create_collection('doc-example-collection-demo')
    print(f"Created collection {collection.collection_id}:")
    pprint(collection.describe_collection())

    print("Indexing faces from three images:")
    for image in images:
        collection.index_faces(image, 10)
    print("Listing faces in collection:")
    faces = collection.list_faces(10)
    for face in faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    print(f"Searching for faces in the collection that match the first face in the "
          f"list (Face ID: {faces[0].face_id}.")
```

```
found_faces = collection.search_faces(faces[0].face_id, 80, 10)
print(f"Found {len(found_faces)} matching faces.")
for face in found_faces:
    pprint(face.to_dict())
input("Press Enter to continue.")

print(f"Searching for faces in the collection that match the largest face in "
      f"{images[0].image_name}.")
image_face, match_faces = collection.search_faces_by_image(images[0], 80, 10)
print(f"The largest face in {images[0].image_name} is:")
pprint(image_face.to_dict())
print(f"Found {len(match_faces)} matching faces.")
for face in match_faces:
    pprint(face.to_dict())
input("Press Enter to continue.")

collection.delete_collection()
print('Thanks for watching!')
print('*'*88)
```

## Detect and display elements in images

The following code example shows how to:

- Detect elements in images by using Amazon Rekognition.
- Display images and draw bounding boxes around detected elements.

For more information, see [Displaying bounding boxes](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create classes to wrap Amazon Rekognition functions.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests

from rekognition_objects import (
    RekognitionFace, RekognitionCelebrity, RekognitionLabel,
    RekognitionModerationLabel, RekognitionText, show_bounding_boxes, show_polygons)

logger = logging.getLogger(__name__)

class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """
    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        
```

```
:param image_name: The name of the image.
:param rekognition_client: A Boto3 Rekognition client.
"""
self.image = image
self.image_name = image_name
self.rekognition_client = rekognition_client

@classmethod
def from_file(cls, image_file_name, rekognition_client, image_name=None):
"""
Creates a RekognitionImage object from a local file.

:param image_file_name: The file name of the image. The file is opened and its
bytes are read.
:param rekognition_client: A Boto3 Rekognition client.
:param image_name: The name of the image. If this is not specified, the
file name is used as the image name.
:return: The RekognitionImage object, initialized with image bytes from the
file.
"""
with open(image_file_name, 'rb') as img_file:
    image = {'Bytes': img_file.read()}
name = image_file_name if image_name is None else image_name
return cls(image, name, rekognition_client)

@classmethod
def from_bucket(cls, s3_object, rekognition_client):
"""
Creates a RekognitionImage object from an Amazon S3 object.

:param s3_object: An Amazon S3 object that identifies the image. The image
is not retrieved until needed for a later call.
:param rekognition_client: A Boto3 Rekognition client.
:return: The RekognitionImage object, initialized with Amazon S3 object data.
"""
image = {'S3Object': {'Bucket': s3_object.bucket_name, 'Name': s3_object.key}}
return cls(image, s3_object.key, rekognition_client)

def detect_faces(self):
"""
Detects faces in the image.

:return: The list of faces found in the image.
"""
try:
    response = self.rekognition_client.detect_faces(
        Image=self.image, Attributes=['ALL'])
    faces = [RekognitionFace(face) for face in response['FaceDetails']]
    logger.info("Detected %s faces.", len(faces))
except ClientError:
    logger.exception("Couldn't detect faces in %s.", self.image_name)
    raise
else:
    return faces

def detect_labels(self, max_labels):
"""
Detects labels in the image. Labels are objects and people.

:param max_labels: The maximum number of labels to return.
:return: The list of labels detected in the image.
"""
try:
    response = self.rekognition_client.detect_labels(
        Image=self.image, MaxLabels=max_labels)
    labels = [RekognitionLabel(label) for label in response['Labels']]

```

```
        logger.info("Found %s labels in %s.", len(labels), self.image_name)
    except ClientError:
        logger.info("Couldn't detect labels in %s.", self.image_name)
        raise
    else:
        return labels

def recognize_celebrities(self):
    """
    Detects celebrities in the image.

    :return: A tuple. The first element is the list of celebrities found in
            the image. The second element is the list of faces that were
            detected but did not match any known celebrities.
    """
    try:
        response = self.rekognition_client.recognize_celebrities(
            Image=self.image)
        celebrities = [RekognitionCelebrity(celeb)
                       for celeb in response['CelebrityFaces']]
        other_faces = [RekognitionFace(face)
                       for face in response['UnrecognizedFaces']]
        logger.info(
            "Found %s celebrities and %s other faces in %s.", len(celebrities),
            len(other_faces), self.image_name)
    except ClientError:
        logger.exception("Couldn't detect celebrities in %s.", self.image_name)
        raise
    else:
        return celebrities, other_faces

def compare_faces(self, target_image, similarity):
    """
    Compares faces in the image with the largest face in the target image.

    :param target_image: The target image to compare against.
    :param similarity: Faces in the image must have a similarity value greater
                       than this value to be included in the results.
    :return: A tuple. The first element is the list of faces that match the
            reference image. The second element is the list of faces that have
            a similarity value below the specified threshold.
    """
    try:
        response = self.rekognition_client.compare_faces(
            SourceImage=self.image,
            TargetImage=target_image.image,
            SimilarityThreshold=similarity)
        matches = [RekognitionFace(match['Face']) for match
                   in response['FaceMatches']]
        unmatches = [RekognitionFace(face) for face in response['UnmatchedFaces']]
        logger.info(
            "Found %s matched faces and %s unmatched faces.",
            len(matches), len(unmatches))
    except ClientError:
        logger.exception(
            "Couldn't match faces from %s to %s.", self.image_name,
            target_image.image_name)
        raise
    else:
        return matches, unmatches

def detect_moderation_labels(self):
    """
    Detects moderation labels in the image. Moderation labels identify content
    that may be inappropriate for some audiences.
    
```

```
:return: The list of moderation labels found in the image.  
"""  
try:  
    response = self.rekognition_client.detect_moderation_labels(  
        Image=self.image)  
    labels = [RekognitionModerationLabel(label)  
              for label in response['ModerationLabels']]  
    logger.info(  
        "Found %s moderation labels in %s.", len(labels), self.image_name)  
except ClientError:  
    logger.exception(  
        "Couldn't detect moderation labels in %s.", self.image_name)  
    raise  
else:  
    return labels  
  
def detect_text(self):  
    """  
    Detects text in the image.  
  
    :return: The list of text elements found in the image.  
    """  
    try:  
        response = self.rekognition_client.detect_text(Image=self.image)  
        texts = [RekognitionText(text) for text in response['TextDetections']]  
        logger.info("Found %s texts in %s.", len(texts), self.image_name)  
    except ClientError:  
        logger.exception("Couldn't detect text in %s.", self.image_name)  
        raise  
    else:  
        return texts
```

Create helper functions to draw bounding boxes and polygons.

```
import io  
import logging  
from PIL import Image, ImageDraw  
  
logger = logging.getLogger(__name__)  
  
def show_bounding_boxes(image_bytes, box_sets, colors):  
    """  
    Draws bounding boxes on an image and shows it with the default image viewer.  
  
    :param image_bytes: The image to draw, as bytes.  
    :param box_sets: A list of lists of bounding boxes to draw on the image.  
    :param colors: A list of colors to use to draw the bounding boxes.  
    """  
    image = Image.open(io.BytesIO(image_bytes))  
    draw = ImageDraw.Draw(image)  
    for boxes, color in zip(box_sets, colors):  
        for box in boxes:  
            left = image.width * box['Left']  
            top = image.height * box['Top']  
            right = (image.width * box['Width']) + left  
            bottom = (image.height * box['Height']) + top  
            draw.rectangle([left, top, right, bottom], outline=color, width=3)  
    image.show()  
  
def show_polygons(image_bytes, polygons, color):  
    """  
    Draws polygons on an image and shows it with the default image viewer.
```

```
:param image_bytes: The image to draw, as bytes.
:param polygons: The list of polygons to draw on the image.
:param color: The color to use to draw the polygons.
"""
image = Image.open(io.BytesIO(image_bytes))
draw = ImageDraw.Draw(image)
for polygon in polygons:
    draw.polygon([
        (image.width * point['X'], image.height * point['Y']) for point in
polygon],
        outline=color)
image.show()
```

Create classes to parse objects returned by Amazon Rekognition.

```
class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""
    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
            functions.
        :param timestamp: The time when the face was detected, if the face was
            detected in a video.
        """
        self.bounding_box = face.get('BoundingBox')
        self.confidence = face.get('Confidence')
        self.landmarks = face.get('Landmarks')
        self.pose = face.get('Pose')
        self.quality = face.get('Quality')
        age_range = face.get('AgeRange')
        if age_range is not None:
            self.age_range = (age_range.get('Low'), age_range.get('High'))
        else:
            self.age_range = None
        self.smile = face.get('Smile', {}).get('Value')
        self.eyeglasses = face.get('Eyglasses', {}).get('Value')
        self.sunglasses = face.get('Sunglasses', {}).get('Value')
        self.gender = face.get('Gender', {}).get('Value', None)
        self.beard = face.get('Beard', {}).get('Value')
        self.mustache = face.get('Mustache', {}).get('Value')
        self.eyes_open = face.get('EyesOpen', {}).get('Value')
        self.mouth_open = face.get('MouthOpen', {}).get('Value')
        self.emotions = [emo.get('Type') for emo in face.get('Emotions', [])]
            if emo.get('Confidence', 0) > 50]
        self.face_id = face.get('FaceId')
        self.image_id = face.get('ImageId')
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
        rendering = {}
        if self.bounding_box is not None:
            rendering['bounding_box'] = self.bounding_box
        if self.age_range is not None:
            rendering['age'] = f'{self.age_range[0]} - {self.age_range[1]}'
        if self.gender is not None:
            rendering['gender'] = self.gender
        if self.emotions:
```

```

        rendering['emotions'] = self.emotions
    if self.face_id is not None:
        rendering['face_id'] = self.face_id
    if self.image_id is not None:
        rendering['image_id'] = self.image_id
    if self.timestamp is not None:
        rendering['timestamp'] = self.timestamp
    has = []
    if self.smile:
        has.append('smile')
    if self.eyeglasses:
        has.append('eyeglasses')
    if self.sunglasses:
        has.append('sunglasses')
    if self.beard:
        has.append('beard')
    if self.mustache:
        has.append('mustache')
    if self.eyes_open:
        has.append('open eyes')
    if self.mouth_open:
        has.append('open mouth')
    if has:
        rendering['has'] = has
    return rendering

class RekognitionCelebrity:
    """Encapsulates an Amazon Rekognition celebrity."""
    def __init__(self, celebrity, timestamp=None):
        """
        Initializes the celebrity object.

        :param celebrity: Celebrity data, in the format returned by Amazon Rekognition
                           functions.
        :param timestamp: The time when the celebrity was detected, if the celebrity
                           was detected in a video.
        """
        self.info_urls = celebrity.get('Urls')
        self.name = celebrity.get('Name')
        self.id = celebrity.get('Id')
        self.face = RekognitionFace(celebrity.get('Face'))
        self.confidence = celebrity.get('MatchConfidence')
        self.bounding_box = celebrity.get('BoundingBox')
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the celebrity data to a dict.

        :return: A dict that contains the celebrity data.
        """
        rendering = self.face.to_dict()
        if self.name is not None:
            rendering['name'] = self.name
        if self.info_urls:
            rendering['info URLs'] = self.info_urls
        if self.timestamp is not None:
            rendering['timestamp'] = self.timestamp
        return rendering

class RekognitionPerson:
    """Encapsulates an Amazon Rekognition person."""
    def __init__(self, person, timestamp=None):
        """
        Initializes the person object.
        
```

```
:param person: Person data, in the format returned by Amazon Rekognition
    functions.
:param timestamp: The time when the person was detected, if the person
    was detected in a video.
"""
self.index = person.get('Index')
self.bounding_box = person.get('BoundingBox')
face = person.get('Face')
self.face = RekognitionFace(face) if face is not None else None
self.timestamp = timestamp

def to_dict(self):
"""
Renders some of the person data to a dict.

:return: A dict that contains the person data.
"""
rendering = self.face.to_dict() if self.face is not None else {}
if self.index is not None:
    rendering['index'] = self.index
if self.bounding_box is not None:
    rendering['bounding_box'] = self.bounding_box
if self.timestamp is not None:
    rendering['timestamp'] = self.timestamp
return rendering

class RekognitionLabel:
    """Encapsulates an Amazon Rekognition label."""
    def __init__(self, label, timestamp=None):
        """
Initializes the label object.

:param label: Label data, in the format returned by Amazon Rekognition
    functions.
:param timestamp: The time when the label was detected, if the label
    was detected in a video.
"""
        self.name = label.get('Name')
        self.confidence = label.get('Confidence')
        self.instances = label.get('Instances')
        self.parents = label.get('Parents')
        self.timestamp = timestamp

    def to_dict(self):
        """
Renders some of the label data to a dict.

:return: A dict that contains the label data.
"""
        rendering = {}
        if self.name is not None:
            rendering['name'] = self.name
        if self.timestamp is not None:
            rendering['timestamp'] = self.timestamp
        return rendering

class RekognitionModerationLabel:
    """Encapsulates an Amazon Rekognition moderation label."""
    def __init__(self, label, timestamp=None):
        """
Initializes the moderation label object.

:param label: Label data, in the format returned by Amazon Rekognition
    functions.
:param timestamp: The time when the moderation label was detected, if the
    label was detected in a video.

```

```

"""
    self.name = label.get('Name')
    self.confidence = label.get('Confidence')
    self.parent_name = label.get('ParentName')
    self.timestamp = timestamp

def to_dict(self):
"""
    Renders some of the moderation label data to a dict.

:return: A dict that contains the moderation label data.
"""
    rendering = {}
    if self.name is not None:
        rendering['name'] = self.name
    if self.parent_name is not None:
        rendering['parent_name'] = self.parent_name
    if self.timestamp is not None:
        rendering['timestamp'] = self.timestamp
    return rendering

class RekognitionText:
"""Encapsulates an Amazon Rekognition text element."""
def __init__(self, text_data):
"""
    Initializes the text object.

:param text_data: Text data, in the format returned by Amazon Rekognition
functions.
"""
    self.text = text_data.get('DetectedText')
    self.kind = text_data.get('Type')
    self.id = text_data.get('Id')
    self.parent_id = text_data.get('ParentId')
    self.confidence = text_data.get('Confidence')
    self.geometry = text_data.get('Geometry')

def to_dict(self):
"""
    Renders some of the text data to a dict.

:return: A dict that contains the text data.
"""
    rendering = {}
    if self.text is not None:
        rendering['text'] = self.text
    if self.kind is not None:
        rendering['kind'] = self.kind
    if self.geometry is not None:
        rendering['polygon'] = self.geometry.get('Polygon')
    return rendering

```

Use the wrapper classes to detect elements in images and display their bounding boxes. The images used in this example can be found on GitHub along with instructions and more code.

```

def usage_demo():
    print('*'*88)
    print("Welcome to the Amazon Rekognition image detection demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    rekognition_client = boto3.client('rekognition')
    street_scene_file_name = ".media/pexels-kaique-rocha-109919.jpg"

```

```
celebrity_file_name = ".media/pexels-pixabay-53370.jpg"
one_girl_url = 'https://dhei5unw3vrsx.cloudfront.net/images/source3_resized.jpg'
three_girls_url = 'https://dhei5unw3vrsx.cloudfront.net/images/target3_resized.jpg'
swimwear_object = boto3.resource('s3').Object(
    'console-sample-images-pdx', 'yoga_swimwear.jpg')
book_file_name = '.media/pexels-christina-morillo-1181671.jpg'

street_scene_image = RekognitionImage.from_file(
    street_scene_file_name, rekognition_client)
print(f"Detecting faces in {street_scene_image.image_name}...")
faces = street_scene_image.detect_faces()
print(f"Found {len(faces)} faces, here are the first three.")
for face in faces[:3]:
    pprint(face.to_dict())
show_bounding_boxes(
    street_scene_image.image['Bytes'], [[face.bounding_box for face in faces]],
    ['aqua'])
input("Press Enter to continue.")

print(f"Detecting labels in {street_scene_image.image_name}...")
labels = street_scene_image.detect_labels(100)
print(f"Found {len(labels)} labels.")
for label in labels:
    pprint(label.to_dict())
names = []
box_sets = []
colors = ['aqua', 'red', 'white', 'blue', 'yellow', 'green']
for label in labels:
    if label.instances:
        names.append(label.name)
        box_sets.append([inst['BoundingBox'] for inst in label.instances])
print(f"Showing bounding boxes for {names} in {colors[:len(names)]}.")
show_bounding_boxes(
    street_scene_image.image['Bytes'], box_sets, colors[:len(names)])
input("Press Enter to continue.")

celebrity_image = RekognitionImage.from_file(
    celebrity_file_name, rekognition_client)
print(f"Detecting celebrities in {celebrity_image.image_name}...")
celebs, others = celebrity_image.recognize_celebrities()
print(f"Found {len(celebs)} celebrities.")
for celeb in celebs:
    pprint(celeb.to_dict())
show_bounding_boxes(
    celebrity_image.image['Bytes'],
    [[celeb.face.bounding_box for celeb in celebs]], ['aqua'])
input("Press Enter to continue.")

girl_image_response = requests.get(one_girl_url)
girl_image = RekognitionImage(
    {'Bytes': girl_image_response.content}, "one-girl", rekognition_client)
group_image_response = requests.get(three_girls_url)
group_image = RekognitionImage(
    {'Bytes': group_image_response.content}, "three-girls", rekognition_client)
print("Comparing reference face to group of faces...")
matches, unmatches = girl_image.compare_faces(group_image, 80)
print(f"Found {len(matches)} face matching the reference face.")
show_bounding_boxes(
    group_image.image['Bytes'], [[match.bounding_box for match in matches]],
    ['aqua'])
input("Press Enter to continue.")

swimwear_image = RekognitionImage.from_bucket(swimwear_object, rekognition_client)
print(f"Detecting suggestive content in {swimwear_object.key}...")
labels = swimwear_image.detect_moderation_labels()
print(f"Found {len(labels)} moderation labels.")
```

```
for label in labels:
    pprint(label.to_dict())
input("Press Enter to continue.")

book_image = RekognitionImage.from_file(book_file_name, rekognition_client)
print(f"Detecting text in {book_image.image_name}...")
texts = book_image.detect_text()
print(f"Found {len(texts)} text instances. Here are the first seven:")
for text in texts[:7]:
    pprint(text.to_dict())
show_polygons(
    book_image.image['Bytes'], [text.geometry['Polygon'] for text in texts],
    'aqua')

print("Thanks for watching!")
print('*'*88)
```

## Amazon S3 examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Simple Storage Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Get started

#### Hello Amazon S3

The following code example shows how to get started using Amazon Simple Storage Service (Amazon S3).

#### Python

##### SDK for Python (Boto3)

###### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import boto3

def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) resource and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """
    s3_resource = boto3.resource('s3')
    print("Hello, Amazon S3! Let's list your buckets:")
    for bucket in s3_resource.buckets.all():
        print(f"\t{bucket.name}")

if __name__ == '__main__':
    hello_s3()
```

- For API details, see [ListBuckets](#) in *AWS SDK for Python (Boto3) API Reference*.

## Topics

- [Actions \(p. 4164\)](#)
- [Scenarios \(p. 4185\)](#)

## Actions

### Add CORS rules to a bucket

The following code example shows how to add cross-origin resource sharing (CORS) rules to an S3 bucket.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3  
                      that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def put_cors(self, cors_rules):  
        """  
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that are  
        allowed from other domains.  
  
        :param cors_rules: The CORS rules to apply.  
        """  
        try:  
            self.bucket.Cors().put(CORSConfiguration={'CORSRules': cors_rules})  
            logger.info(  
                "Put CORS rules %s for bucket '%s'.", cors_rules, self.bucket.name)  
        except ClientError:  
            logger.exception("Couldn't put CORS rules for bucket %s.",  
                            self.bucket.name)  
            raise
```

- For API details, see [PutBucketCors](#) in *AWS SDK for Python (Boto3) API Reference*.

### Add a lifecycle configuration to a bucket

The following code example shows how to add a lifecycle configuration to an S3 bucket.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3  
                      that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def put_lifecycle_configuration(self, lifecycle_rules):  
        """  
        Apply a lifecycle configuration to the bucket. The lifecycle configuration can  
        be used to archive or delete the objects in the bucket according to specified  
        parameters, such as a number of days.  
  
        :param lifecycle_rules: The lifecycle rules to apply.  
        """  
        try:  
            self.bucket.LifecycleConfiguration().put(  
                LifecycleConfiguration={'Rules': lifecycle_rules})  
            logger.info(  
                "Put lifecycle rules %s for bucket '%s'.", lifecycle_rules,  
                self.bucket.name)  
        except ClientError:  
            logger.exception(  
                "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name)  
            raise
```

- For API details, see [PutBucketLifecycleConfiguration](#) in *AWS SDK for Python (Boto3) API Reference*.

## Add a policy to a bucket

The following code example shows how to add a policy to an S3 bucket.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3  
                      that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def put_policy(self, policy):  
        """  
        Apply a security policy to the bucket. Policies control users' ability  
        to perform specific actions, such as listing the objects in the bucket.  
  
        :param policy: The policy to apply to the bucket.  
        """  
        try:
```

```
        self.bucket.Policy().put(Policy=json.dumps(policy))
        logger.info("Put policy %s for bucket '%s'.", policy, self.bucket.name)
    except ClientError:
        logger.exception("Couldn't apply policy to bucket '%s'.", self.bucket.name)
        raise
```

- For API details, see [PutBucketPolicy in AWS SDK for Python \(Boto3\) API Reference](#).

## Copy an object from one bucket to another

The following code example shows how to copy an S3 object from one bucket to another.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
        Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and key.
                           This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(CopySource={
                'Bucket': self.object.bucket_name,
                'Key': self.object.key
            })
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name, self.object.key,
                dest_object.bucket_name, dest_object.key)
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name, self.object.key,
                dest_object.bucket_name, dest_object.key)
            raise
```

- For API details, see [CopyObject in AWS SDK for Python \(Boto3\) API Reference](#).

## Create a bucket

The following code example shows how to create an S3 bucket.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a bucket with default settings.

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def create(self, region_override=None):
        """
        Create an Amazon S3 bucket in the default Region for the account or in the
        specified Region.

        :param region_override: The Region in which to create the bucket. If this is
                               not specified, the Region configured in your shared
                               credentials is used.
        """
        if region_override is not None:
            region = region_override
        else:
            region = self.bucket.meta.client.meta.region_name
        try:
            self.bucket.create(
                CreateBucketConfiguration={'LocationConstraint': region})

            self.bucket.wait_until_exists()
            logger.info(
                "Created bucket '%s' in region=%s", self.bucket.name, region)
        except ClientError as error:
            logger.exception(
                "Couldn't create bucket named '%s' in region=%s.",
                self.bucket.name, region)
            raise error
```

Create a versioned bucket with a lifecycle configuration.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
```

```

try:
    bucket = s3.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            'LocationConstraint': s3.meta.client.meta.region_name
        }
    )
    logger.info("Created bucket %s.", bucket.name)
except ClientError as error:
    if error.response['Error']['Code'] == 'BucketAlreadyOwnedByYou':
        logger.warning("Bucket %s already exists! Using it.", bucket_name)
        bucket = s3.Bucket(bucket_name)
    else:
        logger.exception("Couldn't create bucket %s.", bucket_name)
        raise

try:
    bucket.Versioning().enable()
    logger.info("Enabled versioning on bucket %s.", bucket.name)
except ClientError:
    logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
    raise

try:
    expiration = 7
    bucket.LifecycleConfiguration().put(
        LifecycleConfiguration={
            'Rules': [
                {
                    'Status': 'Enabled',
                    'Prefix': prefix,
                    'NoncurrentVersionExpiration': {'NoncurrentDays': expiration}
                }
            ]
        }
    )
    logger.info("Configured lifecycle to expire noncurrent versions after %s days "
               "on bucket %s.", expiration, bucket.name)
except ClientError as error:
    logger.warning("Couldn't configure lifecycle on bucket %s because %s. "
                  "Continuing anyway.", bucket.name, error)

return bucket

```

- For API details, see [CreateBucket](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete CORS rules from a bucket

The following code example shows how to delete CORS rules from an S3 bucket.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3
                       that wraps bucket actions in a class-like structure.
        """

```

```
    self.bucket = bucket
    self.name = bucket.name

def delete_cors(self):
    """
    Delete the CORS rules from the bucket.

    :param bucket_name: The name of the bucket to update.
    """
    try:
        self.bucket.Cors().delete()
        logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
    except ClientError:
        logger.exception("Couldn't delete CORS from bucket '%s'.",
                         self.bucket.name)
        raise
```

- For API details, see [DeleteBucketCors](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a policy from a bucket

The following code example shows how to delete a policy from an S3 bucket.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3
                      that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_policy(self):
        """
        Delete the security policy from the bucket.
        """
        try:
            self.bucket.Policy().delete()
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete policy for bucket '%s'.",
                             self.bucket.name)
            raise
```

- For API details, see [DeleteBucketPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an empty bucket

The following code example shows how to delete an empty S3 bucket.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3  
                      that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def delete(self):  
        """  
        Delete the bucket. The bucket must be empty or an error is raised.  
        """  
        try:  
            self.bucket.delete()  
            self.bucket.wait_until_not_exists()  
            logger.info("Bucket %s successfully deleted.", self.bucket.name)  
        except ClientError:  
            logger.exception("Couldn't delete bucket %s.", self.bucket.name)  
            raise
```

- For API details, see [DeleteBucket](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an object

The following code example shows how to delete an S3 object.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete an object.

```
class ObjectWrapper:  
    """Encapsulates S3 object actions."""  
    def __init__(self, s3_object):  
        """  
        :param s3_object: A Boto3 Object resource. This is a high-level resource in  
                         Boto3  
                         that wraps object actions in a class-like structure.  
        """  
        self.object = s3_object  
        self.key = self.object.key  
  
    def delete(self):  
        """  
        Deletes the object.  
        """  
        try:  
            self.object.delete()  
            self.object.wait_until_not_exists()  
            logger.info(
```

```
        "Deleted object '%s' from bucket '%s'.",
        self.object.key, self.object.bucket_name)
except ClientError:
    logger.exception(
        "Couldn't delete object '%s' from bucket '%s'.",
        self.object.key, self.object.bucket_name)
    raise
```

Roll an object back to a previous version by deleting later versions of the object.

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(bucket.object_versions.filter(Prefix=object_key),
                      key=attrgetter('last_modified'), reverse=True)

    logger.debug(
        "Got versions:\n%s",
        '\n'.join([f"\t{version.version_id}, last modified {version.last_modified}"
                  for version in versions]))

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(f"{version_id} was not found in the list of versions for "
                      f"{object_key}.")
```

Revive a deleted object by removing the object's active delete marker.

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete marker.
    By removing the delete marker, we make the previous version the latest version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
```

```

response = s3.meta.client.list_object_versions(
    Bucket=bucket.name, Prefix=object_key, MaxKeys=1)

if 'DeleteMarkers' in response:
    latest_version = response['DeleteMarkers'][0]
    if latest_version['IsLatest']:
        logger.info("Object %s was indeed deleted on %s. Let's revive it.",
                    object_key, latest_version['LastModified'])
        obj = bucket.Object(object_key)
        obj.Version(latest_version['VersionId']).delete()
        logger.info("Revived %s, active version is now %s with body '%s'",
                    object_key, obj.version_id, obj.get()['Body'].read())
    else:
        logger.warning("Delete marker is not the latest version for %s!",
                      object_key)
elif 'Versions' in response:
    logger.warning("Got an active version for %s, nothing to do.", object_key)
else:
    logger.error("Couldn't get any version info for %s.", object_key)

```

Create a Lambda handler that removes a delete marker from an S3 object. This handler can be used to efficiently clean up extraneous delete markers in a versioned bucket.

```

import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel('INFO')

s3 = boto3.client('s3')


def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
            operation. When the result code is TemporaryFailure, S3 retries the
            operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event['invocationId']
    invocation_schema_version = event['invocationSchemaVersion']

    results = []
    result_code = None
    result_string = None

    task = event['tasks'][0]
    task_id = task['taskId']

    try:
        obj_key = parse.unquote(task['s3Key'], encoding='utf-8')
        obj_version_id = task['s3VersionId']
        bucket_name = task['s3BucketArn'].split(':')[0]
        logger.info("Got task: remove delete marker %s from object %s.",
                    obj_version_id, obj_key)

```

```

try:
    # If this call does not raise an error, the object version is not a delete
    # marker and should not be deleted.
    response = s3.head_object(
        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id)
    result_code = 'PermanentFailure'
    result_string = f"Object {obj_key}, ID {obj_version_id} is not " \
                    f"a delete marker."

    logger.debug(response)
    logger.warning(result_string)
except ClientError as error:
    delete_marker = error.response['ResponseMetadata']['HTTPHeaders'] \
        .get('x-amz-delete-marker', 'false')
    if delete_marker == 'true':
        logger.info("Object %s, version %s is a delete marker.",
                    obj_key, obj_version_id)
        try:
            s3.delete_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id)
            result_code = 'Succeeded'
            result_string = f"Successfully removed delete marker " \
                            f"{obj_version_id} from object {obj_key}."
            logger.info(result_string)
        except ClientError as error:
            # Mark request timeout as a temporary failure so it will be
            retried.
            if error.response['Error']['Code'] == 'RequestTimeout':
                result_code = 'TemporaryFailure'
                result_string = f"Attempt to remove delete marker from " \
                                f"object {obj_key} timed out."
                logger.info(result_string)
            else:
                raise
        else:
            raise ValueError(f"The x-amz-delete-marker header is either not "
                            f"present or is not 'true'.")
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = 'PermanentFailure'
        result_string = str(error)
        logger.exception(error)
finally:
    results.append({
        'taskId': task_id,
        'resultCode': result_code,
        'resultString': result_string
    })
return {
    'invocationSchemaVersion': invocation_schema_version,
    'treatMissingKeysAs': 'PermanentFailure',
    'invocationId': invocation_id,
    'results': results
}

```

- For API details, see [DeleteObject](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete multiple objects

The following code example shows how to delete multiple objects from an S3 bucket.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Delete a set of objects by using a list of object keys.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
        Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    @staticmethod
    def delete_objects(bucket, object_keys):
        """
        Removes a list of objects from a bucket.
        This operation is done as a batch in a single request.

        :param bucket: The bucket that contains the objects. This is a Boto3 Bucket
                       resource.
        :param object_keys: The list of keys that identify the objects to remove.
        :return: The response that contains data about which objects were deleted
                and any that could not be deleted.
        """
        try:
            response = bucket.delete_objects(Delete={
                'Objects': [
                    {
                        'Key': key
                    } for key in object_keys]
            })
            if 'Deleted' in response:
                logger.info(
                    "Deleted objects '%s' from bucket '%s'.",
                    [del_obj['Key'] for del_obj in response['Deleted']], bucket.name)
            if 'Errors' in response:
                logger.warning(
                    "Could not delete objects '%s' from bucket '%s'.",
                    [{"del_obj['Key']: {del_obj['Code']}"} for del_obj in response['Errors']],
                    bucket.name)
        except ClientError:
            logger.exception("Couldn't delete any objects from bucket %s.",
                             bucket.name)
            raise
        else:
            return response
```

Delete all objects in a bucket.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
        Boto3
                           that wraps object actions in a class-like structure.
```

```
"""
    self.object = s3_object
    self.key = self.object.key

@staticmethod
def empty_bucket(bucket):
    """
        Remove all objects from a bucket.

    :param bucket: The bucket to empty. This is a Boto3 Bucket resource.
    """
    try:
        bucket.objects.delete()
        logger.info("Emptied bucket '%s'.", bucket.name)
    except ClientError:
        logger.exception("Couldn't empty bucket '%s'.", bucket.name)
        raise
```

Permanently delete a versioned object by deleting all of its versions.

```
def permanently_delete_object(bucket, object_key):
    """
        Permanently deletes a versioned object by deleting all of its versions.

        Usage is shown in the usage_demo_single_object function at the end of this module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

- For API details, see [DeleteObjects](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete the lifecycle configuration of a bucket

The following code example shows how to delete the lifecycle configuration of an S3 bucket.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:
    """
        Encapsulates S3 bucket actions.
    """
    def __init__(self, bucket):
        """
            :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3
                           that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_lifecycle_configuration(self):
```

```
"""
Remove the lifecycle configuration from the specified bucket.
"""
try:
    self.bucket.LifecycleConfiguration().delete()
    logger.info(
        "Deleted lifecycle configuration for bucket '%s'.", self.bucket.name)
except ClientError:
    logger.exception(
        "Couldn't delete lifecycle configuration for bucket '%s'.",
        self.bucket.name)
raise
```

- For API details, see [DeleteBucketLifecycle](#) in *AWS SDK for Python (Boto3) API Reference*.

## Determine the existence of a bucket

The following code example shows how to determine the existence of an S3 bucket.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3
                      that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def exists(self):
        """
        Determine whether the bucket exists and you have access to it.

        :return: True when the bucket exists; otherwise, False.
        """
        try:
            self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
            logger.info("Bucket %s exists.", self.bucket.name)
            exists = True
        except ClientError:
            logger.warning(
                "Bucket %s doesn't exist or you don't have access to it.",
                self.bucket.name)
            exists = False
        return exists
```

- For API details, see [HeadBucket](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get CORS rules for a bucket

The following code example shows how to get cross-origin resource sharing (CORS) rules for an S3 bucket.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3  
                      that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def get_cors(self):  
        """  
        Get the CORS rules for the bucket.  
  
        :return The CORS rules for the specified bucket.  
        """  
        try:  
            cors = self.bucket.Cors()  
            logger.info(  
                "Got CORS rules %s for bucket '%s'.", cors.cors_rules,  
                self.bucket.name)  
        except ClientError:  
            logger.exception(("Couldn't get CORS for bucket %s.", self.bucket.name))  
        else:  
            return cors
```

- For API details, see [GetBucketCors](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get an object from a bucket

The following code example shows how to read data from an object in an S3 bucket.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ObjectWrapper:  
    """Encapsulates S3 object actions."""  
    def __init__(self, s3_object):  
        """  
        :param s3_object: A Boto3 Object resource. This is a high-level resource in  
                         Boto3  
                         that wraps object actions in a class-like structure.  
        """  
        self.object = s3_object  
        self.key = self.object.key  
  
    def get(self):  
        """  
        Gets the object.
```

```
:return: The object data in bytes.  
"""  
try:  
    body = self.object.get()['Body'].read()  
    logger.info(  
        "Got object '%s' from bucket '%s'.",  
        self.object.key, self.object.bucket_name)  
except ClientError:  
    logger.exception(  
        "Couldn't get object '%s' from bucket '%s'.",  
        self.object.key, self.object.bucket_name)  
    raise  
else:  
    return body
```

- For API details, see [GetObject](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get the ACL of a bucket

The following code example shows how to get the access control list (ACL) of an S3 bucket.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3  
                      that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def get_acl(self):  
        """  
        Get the ACL of the bucket.  
  
        :return: The ACL of the bucket.  
        """  
        try:  
            acl = self.bucket.Acl()  
            logger.info(  
                "Got ACL for bucket %s. Owner is %s.", self.bucket.name, acl.owner)  
        except ClientError:  
            logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)  
            raise  
        else:  
            return acl
```

- For API details, see [GetBucketAcl](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get the ACL of an object

The following code example shows how to get the access control list (ACL) of an S3 object.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
        Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def get_acl(self):
        """
        Gets the ACL of the object.

        :return: The ACL of the object.
        """
        try:
            acl = self.object.Acl()
            logger.info(
                "Got ACL for object %s owned by %s.",
                self.object.key, acl.owner['DisplayName'])
        except ClientError:
            logger.exception("Couldn't get ACL for object %s.", self.object.key)
            raise
        else:
            return acl
```

- For API details, see [GetObjectAcl](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get the lifecycle configuration of a bucket

The following code example shows how to get the lifecycle configuration of an S3 bucket.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3
                       that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_lifecycle_configuration(self):
        """
        Get the lifecycle configuration of the bucket.
        
```

```
:return: The lifecycle rules of the specified bucket.  
"""  
try:  
    config = self.bucket.LifecycleConfiguration()  
    logger.info(  
        "Got lifecycle rules %s for bucket '%s'.", config.rules,  
        self.bucket.name)  
except:  
    logger.exception(  
        "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name)  
    raise  
else:  
    return config.rules
```

- For API details, see [GetBucketLifecycleConfiguration](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get the policy for a bucket

The following code example shows how to get the policy for an S3 bucket.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3  
                      that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def get_policy(self):  
        """  
        Get the security policy of the bucket.  
  
        :return: The security policy of the specified bucket, in JSON format.  
        """  
        try:  
            policy = self.bucket.Policy()  
            logger.info("Got policy %s for bucket '%s'.", policy.policy,  
                        self.bucket.name)  
        except ClientError:  
            logger.exception("Couldn't get policy for bucket '%s'.", self.bucket.name)  
            raise  
        else:  
            return json.loads(policy.policy)
```

- For API details, see [GetBucketPolicy](#) in *AWS SDK for Python (Boto3) API Reference*.

## List buckets

The following code example shows how to list S3 buckets.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3  
                      that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    @staticmethod  
    def list(s3_resource):  
        """  
        Get the buckets in all Regions for the current account.  
  
        :param s3_resource: A Boto3 S3 resource. This is a high-level resource in Boto3  
                            that contains collections and factory methods to create  
                            other high-level S3 sub-resources.  
        :return: The list of buckets.  
        """  
        try:  
            buckets = list(s3_resource.buckets.all())  
            logger.info("Got buckets: %s.", buckets)  
        except ClientError:  
            logger.exception("Couldn't get buckets.")  
            raise  
        else:  
            return buckets
```

- For API details, see [ListBuckets](#) in *AWS SDK for Python (Boto3) API Reference*.

## List objects in a bucket

The following code example shows how to list objects in an S3 bucket.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ObjectWrapper:  
    """Encapsulates S3 object actions."""  
    def __init__(self, s3_object):  
        """  
        :param s3_object: A Boto3 Object resource. This is a high-level resource in  
                         Boto3  
                         that wraps object actions in a class-like structure.  
        """  
        self.object = s3_object  
        self.key = self.object.key  
  
    @staticmethod
```

```
def list(bucket, prefix=None):
    """
    Lists the objects in a bucket, optionally filtered by a prefix.

    :param bucket: The bucket to query. This is a Boto3 Bucket resource.
    :param prefix: When specified, only objects that start with this prefix are
    listed.
    :return: The list of objects.
    """
    try:
        if not prefix:
            objects = list(bucket.objects.all())
        else:
            objects = list(bucket.objects.filter(Prefix=prefix))
        logger.info("Got objects %s from bucket '%s'",
                    [o.key for o in objects], bucket.name)
    except ClientError:
        logger.exception("Couldn't get objects for bucket '%s'.", bucket.name)
        raise
    else:
        return objects
```

- For API details, see [ListObjects](#) in *AWS SDK for Python (Boto3) API Reference*.

## Set a new ACL for a bucket

The following code example shows how to set a new access control list (ACL) for an S3 bucket.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3
                      that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def grant_log_delivery_access(self):
        """
        Grant the AWS Log Delivery group write access to the bucket so that
        Amazon S3 can deliver access logs to the bucket. This is the only recommended
        use of an S3 bucket ACL.
        """
        try:
            acl = self.bucket.Acl()
            # Putting an ACL overwrites the existing ACL. If you want to preserve
            # existing grants, append new grants to the list of existing grants.
            grants = acl.grants if acl.grants else []
            grants.append({
                'Grantee': {
                    'Type': 'Group',
                    'URI': 'http://acs.amazonaws.com/groups/s3/LogDelivery'
                },
                'Permission': 'WRITE'
```

```
        })
        acl.put(
            AccessControlPolicy={
                'Grants': grants,
                'Owner': acl.owner
            }
        )
        logger.info("Granted log delivery access to bucket '%s'", self.bucket.name)
    except ClientError:
        logger.exception("Couldn't add ACL to bucket '%s'.", self.bucket.name)
        raise
```

- For API details, see [PutBucketAcl](#) in *AWS SDK for Python (Boto3) API Reference*.

## Set the ACL of an object

The following code example shows how to set the access control list (ACL) of an S3 object.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
        Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def put_acl(self, email):
        """
        Applies an ACL to the object that grants read access to an AWS user identified
        by email address.

        :param email: The email address of the user to grant access.
        """
        try:
            acl = self.object.Acl()
            # Putting an ACL overwrites the existing ACL, so append new grants
            # if you want to preserve existing grants.
            grants = acl.grants if acl.grants else []
            grants.append({
                'Grantee': {
                    'Type': 'AmazonCustomerByEmail',
                    'EmailAddress': email
                },
                'Permission': 'READ'
            })
            acl.put(
                AccessControlPolicy={
                    'Grants': grants,
                    'Owner': acl.owner
                }
            )
            logger.info("Granted read access to %s.", email)
        except ClientError:
            logger.exception("Couldn't add ACL to object '%s'.", self.key)
```

```
        except ClientError:
            logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
            raise
```

- For API details, see [PutObjectAcl](#) in *AWS SDK for Python (Boto3) API Reference*.

## Upload an object to a bucket

The following code example shows how to upload an object to an S3 bucket.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""
    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
        Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def put(self, data):
        """
        Upload data to the object.

        :param data: The data to upload. This can either be bytes or a string. When
        this
                           argument is a string, it is interpreted as a file name, which is
                           opened in read bytes mode.
        """
        put_data = data
        if isinstance(data, str):
            try:
                put_data = open(data, 'rb')
            except IOError:
                logger.exception("Expected file name or binary data, got '%s'.", data)
                raise

        try:
            self.object.put(Body=put_data)
            self.object.wait_until_exists()
            logger.info(
                "Put object '%s' to bucket '%s'.", self.object.key,
                self.object.bucket_name)
        except ClientError:
            logger.exception(
                "Couldn't put object '%s' to bucket '%s'.", self.object.key,
                self.object.bucket_name)
            raise
        finally:
            if getattr(put_data, 'close', None):
                put_data.close()
```

- For API details, see [PutObject](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Create a presigned URL

The following code example shows how to create a presigned URL for S3 and upload an object.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Generate a presigned URL that can perform an S3 action for a limited time. Use the Requests package to make a request with the URL.

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)

def generate_presigned_url(s3_client, client_method, method_parameters, expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method,
            Params=method_parameters,
            ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.", client_method)
        raise
    return url

def usage_demo():
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('*'*88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print('*'*88)

    parser = argparse.ArgumentParser()
    parser.add_argument('bucket', help="The name of the bucket.")
    parser.add_argument(
        'key', help="For a GET operation, the key of the object in Amazon S3. For a "
                    "PUT operation, the name of a file to upload.")
    parser.add_argument(
        'action', choices=('get', 'put'), help="The action to perform.")
    args = parser.parse_args()
```

```

s3_client = boto3.client('s3')
client_action = 'get_object' if args.action == 'get' else 'put_object'
url = generate_presigned_url(
    s3_client, client_action, {'Bucket': args.bucket, 'Key': args.key}, 1000)

print("Using the Requests package to send a request to the URL.")
response = None
if args.action == 'get':
    response = requests.get(url)
elif args.action == 'put':
    print("Putting data to the URL.")
    try:
        with open(args.key, 'r') as object_file:
            object_text = object_file.read()
        response = requests.put(url, data=object_text)
    except FileNotFoundError:
        print(f"Couldn't find {args.key}. For a PUT operation, the key must be the"
        "name of a file that exists on your computer.")

if response is not None:
    print("Got response:")
    print(f"Status: {response.status_code}")
    print(response.text)

print('*'*88)

if __name__ == '__main__':
    usage_demo()

```

Generate a presigned POST request to upload a file.

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""
    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in Boto3
                      that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def generate_presigned_post(self, object_key, expires_in):
        """
        Generate a presigned Amazon S3 POST request to upload a file.
        A presigned POST can be used for a limited time to let someone without an AWS
        account upload a file to a bucket.

        :param object_key: The object key to identify the uploaded object.
        :param expires_in: The number of seconds the presigned POST is valid.
        :return: A dictionary that contains the URL and form fields that contain
                required access data.
        """
        try:
            response = self.bucket.meta.client.generate_presigned_post(
                Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in)
            logger.info("Got presigned POST URL: %s", response['url'])
        except ClientError:
            logger.exception(
                "Couldn't get a presigned POST URL for bucket '%s' and object '%s'",
                self.bucket.name, object_key)
            raise
        return response

```

## Get started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.
- Delete a bucket.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import io
import os
import uuid

import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError


def do_scenario(s3_resource):
    print('-'*88)
    print("Welcome to the Amazon S3 getting started demo!")
    print('-'*88)

    bucket_name = f'doc-example-bucket-{uuid.uuid4()}'  

    bucket = s3_resource.Bucket(bucket_name)
    try:
        bucket.create(
            CreateBucketConfiguration={
                'LocationConstraint': s3_resource.meta.client.meta.region_name})
        print(f"Created demo bucket named {bucket.name}.")
    except ClientError as err:
        print(f"Tried and failed to create demo bucket {bucket_name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']['Message']}")
        print(f"\nCan't continue the demo without a bucket!")
    return

    file_name = None
    while file_name is None:
        file_name = input("\nEnter a file you want to upload to your bucket: ")
        if not os.path.exists(file_name):
            print(f"Couldn't find file {file_name}. Are you sure it exists?")
        file_name = None

    obj = bucket.Object(os.path.basename(file_name))
    try:
        obj.upload_file(file_name)
        print(f"Uploaded file {file_name} into bucket {bucket.name} with key {obj.key}.")
```

```

        except S3UploadFailedError as err:
            print(f"Couldn't upload file {file_name} to {bucket.name}.")
            print(f"\t{err}")

        answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
        if answer.lower() == 'y':
            data = io.BytesIO()
            try:
                obj.download_fileobj(data)
                data.seek(0)
                print(f"Got your object. Here are the first 20 bytes:\n")
                print(f"\t{data.read(20)}")
            except ClientError as err:
                print(f"Couldn't download {obj.key}.")
                print(f"\t{err.response['Error']['Code']}:{err.response['Error']['Message']}")

        answer = input(
            f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? ")
        if answer.lower() == 'y':
            dest_obj = bucket.Object(f'demo-folder/{obj.key}')
            try:
                dest_obj.copy({'Bucket': bucket.name, 'Key': obj.key})
                print(f"Copied {obj.key} to {dest_obj.key}.")
            except ClientError as err:
                print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
                print(f"\t{err.response['Error']['Code']}:{err.response['Error']['Message']}")

        print("\nYour bucket contains the following objects:")
        try:
            for o in bucket.objects.all():
                print(f"\t{o.key}")
        except ClientError as err:
            print(f"Couldn't list the objects in bucket {bucket.name}.")
            print(f"\t{err.response['Error']['Code']}:{err.response['Error']['Message']}")

        answer = input(
            "\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
        if answer.lower() == 'y':
            try:
                bucket.objects.delete()
                bucket.delete()
                print(f"Emptied and deleted bucket {bucket.name}.\n")
            except ClientError as err:
                print(f"Couldn't empty and delete bucket {bucket.name}.")
                print(f"\t{err.response['Error']['Code']}:{err.response['Error']['Message']}")

        print("Thanks for watching!")
        print('-'*88)

if __name__ == '__main__':
    do_scenario(boto3.resource('s3'))

```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)

- [ListObjects](#)
- [PutObject](#)

## Manage versioned objects in batches with a Lambda function

The following code example shows how to manage versioned S3 objects in batches with a Lambda function.

### SDK for Python (Boto3)

Shows how to manipulate Amazon Simple Storage Service (Amazon S3) versioned objects in batches by creating jobs that call AWS Lambda functions to perform processing. This example creates a version-enabled bucket, uploads the stanzas from the poem *You Are Old, Father William* by Lewis Carroll, and uses Amazon S3 batch jobs to twist the poem in various ways.

#### Learn how to:

- Create Lambda functions that operate on versioned objects.
- Create a manifest of objects to update.
- Create batch jobs that invoke Lambda functions to update objects.
- Delete Lambda functions.
- Empty and delete a versioned bucket.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon S3

## Upload or download large files

The following code example shows how to upload or download large files to and from Amazon S3.

For more information, see [Uploading an object using multipart upload](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that transfer files using several of the available transfer manager settings. Use a callback class to write callback progress during file transfer.

```
import sys
import threading

import boto3
from boto3.s3.transfer import TransferConfig

MB = 1024 * 1024
s3 = boto3.resource('s3')

class TransferCallback:
    """
```

```
Handle callbacks from the transfer manager.

The transfer manager periodically calls the __call__ method throughout
the upload and download process so that it can take action, such as
displaying progress to the user and collecting data about the transfer.
"""

def __init__(self, target_size):
    self._target_size = target_size
    self._total_transferred = 0
    self._lock = threading.Lock()
    self.thread_info = {}

def __call__(self, bytes_transferred):
    """
    The callback method that is called by the transfer manager.

    Display progress during file transfer and collect per-thread transfer
    data. This method can be called by multiple threads, so shared instance
    data is protected by a thread lock.
    """
    thread = threading.current_thread()
    with self._lock:
        self._total_transferred += bytes_transferred
        if thread.ident not in self.thread_info.keys():
            self.thread_info[thread.ident] = bytes_transferred
        else:
            self.thread_info[thread.ident] += bytes_transferred

        target = self._target_size * MB
        sys.stdout.write(
            f"\r{self._total_transferred} of {target} transferred "
            f"({(self._total_transferred / target) * 100:.2f}%).")
        sys.stdout.flush()

def upload_with_default_configuration(local_file_path, bucket_name,
                                      object_key, file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Callback=transfer_callback)
    return transfer_callback.thread_info

def upload_with_chunksize_and_meta(local_file_path, bucket_name, object_key,
                                   file_size_mb, metadata=None):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_chunksize=1 * MB)
```

```
extra_args = {'Metadata': metadata} if metadata else None
s3.Bucket(bucket_name).upload_file(
    local_file_path,
    object_key,
    Config=config,
    ExtraArgs=extra_args,
    Callback=transfer_callback)
return transfer_callback.thread_info

def upload_with_high_threshold(local_file_path, bucket_name, object_key,
                               file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        Callback=transfer_callback)
    return transfer_callback.thread_info

def upload_with_sse(local_file_path, bucket_name, object_key,
                    file_size_mb, sse_key=None):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {
            'SSECustomerAlgorithm': 'AES256',
            'SSECustomerKey': sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        ExtraArgs=extra_args,
        Callback=transfer_callback)
    return transfer_callback.thread_info

def download_with_default_configuration(bucket_name, object_key,
                                         download_file_path, file_size_mb):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path,
        Callback=transfer_callback)
    return transfer_callback.thread_info
```

```
def download_with_single_thread(bucket_name, object_key,
                                 download_file_path, file_size_mb):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path,
        Config=config,
        Callback=transfer_callback)
    return transfer_callback.thread_info

def download_with_high_threshold(bucket_name, object_key,
                                  download_file_path, file_size_mb):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path,
        Config=config,
        Callback=transfer_callback)
    return transfer_callback.thread_info

def download_with_sse(bucket_name, object_key, download_file_path,
                      file_size_mb, sse_key):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {
            'SSECustomerAlgorithm': 'AES256',
            'SSECustomerKey': sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path,
        ExtraArgs=extra_args,
        Callback=transfer_callback)
    return transfer_callback.thread_info
```

Demonstrate the transfer manager functions and report results.

```
import hashlib
import os
```

```
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = ('multipart_threshold', 'multipart_chunksize', 'max_concurrency',
                 'use_threads')
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ('max_io_queue', 'io_chunksize', 'num_download_attempts')

class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource('s3')
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
        self._file_size_mb = 30
        self._demo_folder = None
        self._demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self._demo_folder:
            self._demo_folder = input(
                "Which file folder do you want to use to store "
                "demonstration files? ")
            if not os.path.isdir(self._demo_folder):
                print(f"{self._demo_folder} isn't a folder!")
                self._demo_folder = None

        while not self._demo_bucket:
            self._demo_bucket = input(
                "Which Amazon S3 bucket do you want to use to store "
                "demonstration files? ")
        try:
            self._s3.meta.client.head_bucket(Bucket=self._demo_bucket)
        except ParamValidationError as err:
            print(err)
            self._demo_bucket = None
        except ClientError as err:
            print(err)
            print(
                f"Either {self._demo_bucket} doesn't exist or you don't "
                f"have access to it.")
            self._demo_bucket = None
```

```
def demo(self, question, upload_func, download_func,
         upload_args=None, download_args=None):
    """Run a demonstration.

    Ask the user if they want to run this specific demonstration.
    If they say yes, create a file on the local path, upload it
    using the specified upload function, then download it using the
    specified download function.
    """
    if download_args is None:
        download_args = {}
    if upload_args is None:
        upload_args = {}
    question = question.format(self.file_size_mb)
    answer = input(f"{question} (y/n)")
    if answer.lower() == 'y':
        local_file_path, object_key, download_file_path = \
            self._create_demo_file()

        file_transfer.TransferConfig = \
            self._config_wrapper(TransferConfig, CONFIG_ATTRS)
        self._report_transfer_params('Uploading', local_file_path,
                                      object_key, **upload_args)
        start_time = time.perf_counter()
        thread_info = upload_func(local_file_path, self.demo_bucket,
                                  object_key, self.file_size_mb,
                                  **upload_args)
        end_time = time.perf_counter()
        self._report_transfer_result(thread_info, end_time - start_time)

        file_transfer.TransferConfig = \
            self._config_wrapper(TransferConfig,
                               CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS)
        self._report_transfer_params('Downloading', object_key,
                                     download_file_path, **download_args)
        start_time = time.perf_counter()
        thread_info = download_func(self.demo_bucket, object_key,
                                    download_file_path, self.file_size_mb,
                                    **download_args)
        end_time = time.perf_counter()
        self._report_transfer_result(thread_info, end_time - start_time)

    def last_name_set(self):
        """Get the name set used for the last demo."""
        return self._chore_list[-1]

    def cleanup(self):
        """
        Remove files from the demo folder, and uploaded objects from the
        Amazon S3 bucket.
        """
        print('-' * self._terminal_width)
        for local_file_path, s3_object_key, downloaded_file_path \
                in self._chore_list:
            print(f"Removing {local_file_path}")
            try:
                os.remove(local_file_path)
            except FileNotFoundError as err:
                print(err)

            print(f"Removing {downloaded_file_path}")
            try:
                os.remove(downloaded_file_path)
            except FileNotFoundError as err:
                print(err)
```

```
if self.demo_bucket:
    print(f"Removing {self.demo_bucket}:{s3_object_key}")
    try:
        self._s3.Bucket(self.demo_bucket).Object(
            s3_object_key).delete()
    except ClientError as err:
        print(err)

def _setup_platform_specific(self):
    """Set up platform-specific command used to create a large file."""
    if platform.system() == "Windows":
        self._create_file_cmd = "fsutil file createnew {} {}"
        self._size_multiplier = MB
    elif platform.system() == "Linux" or platform.system() == "Darwin":
        self._create_file_cmd = f"dd if=/dev/urandom of={{}{}} " \
                               f"bs={MB} count={{}}"
        self._size_multiplier = 1
    else:
        raise EnvironmentError(
            f"Demo of platform {platform.system()} isn't supported.")

def _create_demo_file(self):
    """
    Create a file in the demo folder specified by the user. Store the local
    path, object name, and download path for later cleanup.

    Only the local file is created by this method. The Amazon S3 object and
    download file are created later during the demonstration.

    Returns:
    A tuple that contains the local file path, object name, and download
    file path.
    """
    file_name_template = "TestFile{}-{}.demo"
    local_suffix = "local"
    object_suffix = "s3object"
    download_suffix = "downloaded"
    file_tag = len(self._chore_list) + 1

    local_file_path = os.path.join(
        self.demo_folder,
        file_name_template.format(file_tag, local_suffix))

    s3_object_key = file_name_template.format(file_tag, object_suffix)

    downloaded_file_path = os.path.join(
        self.demo_folder,
        file_name_template.format(file_tag, download_suffix))

    filled_cmd = self._create_file_cmd.format(
        local_file_path,
        self.file_size_mb * self._size_multiplier)

    print(f"Creating file of size {self.file_size_mb} MB "
          f"in {self.demo_folder} by running:")
    print(f"{'':>4}{filled_cmd}")
    os.system(filled_cmd)

    chore = (local_file_path, s3_object_key, downloaded_file_path)
    self._chore_list.append(chore)
    return chore

def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
    """Report configuration and extra arguments used for a file transfer."""
    print('-' * self._terminal_width)
```

```
print(f'{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}')
if kwargs:
    print('With extra args:')
    for arg, value in kwargs.items():
        print(f'{"":4}{arg:<20}: {value}')

@staticmethod
def ask_user(question):
    """
    Ask the user a yes or no question.

    Returns:
    True when the user answers 'y' or 'Y'; otherwise, False.
    """
    answer = input(f"{question} (y/n) ")
    return answer.lower() == 'y'

@staticmethod
def _config_wrapper(func, config_attrs):
    def wrapper(*args, **kwargs):
        config = func(*args, **kwargs)
        print('With configuration:')
        for attr in config_attrs:
            print(f'{"":4}{attr:<20}: {getattr(config, attr)}')
        return config

    return wrapper

@staticmethod
def _report_transfer_result(thread_info, elapsed):
    """Report the result of a transfer, including per-thread data."""
    print(f"\nUsed {len(thread_info)} threads.")
    for ident, byte_count in thread_info.items():
        print(f'{"":4}Thread {ident} copied {byte_count} bytes.')
    print(f"Your transfer took {elapsed:.2f} seconds.")

def main():
    """
    Run the demonstration script for s3_file_transfer.

    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration)

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold)

    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and "

```

```

    "then download the same file using a single thread?",  

    file_transfer.upload_with_chunksize_and_meta,  

    file_transfer.download_with_single_thread,  

    upload_args={  

        'metadata': {  

            'upload_type': 'chunky',  

            'favorite_color': 'aqua',  

            'size': 'medium'})}  
  

    # Upload using server-side encryption with customer-provided  

    # encryption keys.  

    # Generate a 256-bit key from a passphrase.  

    sse_key = hashlib.sha256('demo_passphrase'.encode('utf-8')).digest()  

demo_manager.demo(  

    "Do you want to upload and download a {} MB file using "  

    "server-side encryption?",  

    file_transfer.upload_with_sse,  

    file_transfer.download_with_sse,  

    upload_args={'sse_key': sse_key},  

    download_args={'sse_key': sse_key})  
  

    # Download without specifying an encryption key to show that the  

    # encryption key must be included to download an encrypted object.  

if demo_manager.ask_user("Do you want to try to download the encrypted "  

    "object without sending the required key?"):  

    try:  

        _, object_key, download_file_path = \  

            demo_manager.last_name_set()  

        file_transfer.download_with_default_configuration(  

            demo_manager.demo_bucket, object_key, download_file_path,  

            demo_manager.file_size_mb)  

    except ClientError as err:  

        print("Got expected error when trying to download an encrypted "  

            "object without specifying encryption info:")  

        print(f"'{err}'")  
  

    # Remove all created and downloaded files, remove all objects from  

    # S3 storage.  

if demo_manager.ask_user(  

    "Demonstration complete. Do you want to remove local files "  

    "and S3 objects?"):  

    demo_manager.cleanup()  
  

if __name__ == '__main__':  

    try:  

        main()  

    except NoCredentialsError as error:  

        print(error)  

        print("To run this example, you must have valid credentials in "  

            "a shared credential file or set in environment variables.")

```

## Work with versioned objects

The following code example shows how to:

- Create a versioned S3 bucket.
- Get all versions of an object.
- Roll an object back to a previous version.
- Delete and restore a versioned object.
- Permanently delete all versions of an object.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions that wrap S3 actions.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """

    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                'LocationConstraint': s3.meta.client.meta.region_name
            }
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response['Error']['Code'] == 'BucketAlreadyOwnedByYou':
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                'Rules': [
                    {
                        'Status': 'Enabled',
                        'Prefix': prefix,
                        'NoncurrentVersionExpiration': {'NoncurrentDays': expiration}
                    }
                ]
            }
        )
        logger.info("Configured lifecycle to expire noncurrent versions after %s days "
                   "on bucket %s.", expiration, bucket.name)
    except ClientError as error:
        logger.warning("Couldn't configure lifecycle on bucket %s because %s. "
                      "Continuing anyway.", bucket.name, error)

    return bucket
```

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(bucket.object_versions.filter(Prefix=object_key),
                      key=attrgetter('last_modified'), reverse=True)

    logger.debug(
        "Got versions:\n%s",
        '\n'.join([f"\t{version.version_id}, last modified {version.last_modified}"
                  for version in versions]))

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(f"{version_id} was not found in the list of versions for "
                      f"{object_key}.")

def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete marker.
    By removing the delete marker, we make the previous version the latest version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1)

    if 'DeleteMarkers' in response:
        latest_version = response['DeleteMarkers'][0]
        if latest_version['IsLatest']:
            logger.info("Object %s was indeed deleted on %s. Let's revive it.",
                        object_key, latest_version['LastModified'])
            obj = bucket.Object(object_key)
            obj.Version(latest_version['VersionId']).delete()
            logger.info("Revived %s, active version is now %s with body '%s'",
                        object_key, obj.version_id, obj.get()['Body'].read())
        else:
            logger.warning("Delete marker is not the latest version for %s!",
                           object_key)
    elif 'Versions' in response:
```

```

        logger.warning("Got an active version for %s, nothing to do.", object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)

def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise

```

Upload the stanza of a poem to a versioned object and perform a series of actions on it.

```

def usage_demo_single_object(obj_prefix='demo-versioning/'):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open('father_william.txt') as file:
        stanzas = file.read().split('\n\n')

    width = get_terminal_size((80, 20))[0]
    print('-'*width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
    print("This demonstration uploads a single stanza of a poem to an Amazon "
          "S3 bucket and then applies various revisions to it.")
    print('-'*width)
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket('bucket-' + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])

    # Add the first stanza and revise it a few times.
    print("\nApplying some revisions to the stanza...")
    obj_stanza_1 = bucket.Object(f'{obj_prefix}stanza-1')
    obj_stanza_1.put(Body=bytes(stanzas[0], 'utf-8'))
    obj_stanza_1.put(Body=bytes(stanzas[0].upper(), 'utf-8'))
    obj_stanza_1.put(Body=bytes(stanzas[0].lower(), 'utf-8'))
    obj_stanza_1.put(Body=bytes(stanzas[0][::-1], 'utf-8'))
    print("The latest version of the stanza is now:",
          obj_stanza_1.get()['Body'].read().decode('utf-8'),
          sep='\n')

    # Versions are returned in order, most recent first.
    obj_stanza_1_versions = bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    print(
        "The version data of the stanza revisions:",
        *[f"    {version.version_id}, last modified {version.last_modified}"
          for version in obj_stanza_1_versions],
        sep='\n'
    )

```

```
# Rollback two versions.
print("\nRolling back two versions...")
rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
print("The latest version of the stanza:",
      obj_stanza_1.get()['Body'].read().decode('utf-8'),
      sep='\n')

# Delete the stanza
print("\nDeleting the stanza...")
obj_stanza_1.delete()
try:
    obj_stanza_1.get()
except ClientError as error:
    if error.response['Error']['Code'] == 'NoSuchKey':
        print("The stanza is now deleted (as expected).")
    else:
        raise

# Revive the stanza
print("\nRestoring the stanza...")
revive_object(bucket, obj_stanza_1.key)
print("The stanza is restored! The latest version is again:",
      obj_stanza_1.get()['Body'].read().decode('utf-8'),
      sep='\n')

# Permanently delete all versions of the object. This cannot be undone!
print("\nPermanently deleting all versions of the stanza...")
permanently_delete_object(bucket, obj_stanza_1.key)
obj_stanza_1_versions = bucket.object_versions.filter(Prefix=obj_stanza_1.key)
if len(list(obj_stanza_1_versions)) == 0:
    print("The stanza has been permanently deleted and now has no versions.")
else:
    print("Something went wrong. The stanza still exists!")

print(f"\nRemoving {bucket.name}...")
bucket.delete()
print(f"{bucket.name} deleted.")
print("Demo done!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateBucket](#)
  - [DeleteObject](#)
  - [ListObjectVersions](#)
  - [PutBucketLifecycleConfiguration](#)

## S3 Glacier examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon S3 Glacier.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4202\)](#)
- [Scenarios \(p. 4210\)](#)

## Actions

### Create a vault

The following code example shows how to create an Amazon S3 Glacier vault.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:  
    """Encapsulates Amazon S3 Glacier API operations."""  
    def __init__(self, glacier_resource):  
        """  
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.  
        """  
        self.glacier_resource = glacier_resource  
  
    def create_vault(self, vault_name):  
        """  
        Creates a vault.  
  
        :param vault_name: The name to give the vault.  
        :return: The newly created vault.  
        """  
        try:  
            vault = self.glacier_resource.create_vault(vaultName=vault_name)  
            logger.info("Created vault %s.", vault_name)  
        except ClientError:  
            logger.exception("Couldn't create vault %s.", vault_name)  
        else:  
            return vault
```

- For API details, see [CreateVault](#) in *AWS SDK for Python (Boto3) API Reference*.

### Delete a vault

The following code example shows how to delete an Amazon S3 Glacier vault.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:  
    """Encapsulates Amazon S3 Glacier API operations."""  
    def __init__(self, glacier_resource):  
        """  
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.  
        """  
        self.glacier_resource = glacier_resource  
  
    @staticmethod  
    def delete_vault(vault):
```

```
"""
Deletes a vault.

:param vault: The vault to delete.
"""
try:
    vault.delete()
    logger.info("Deleted vault %s.", vault.name)
except ClientError:
    logger.exception("Couldn't delete vault %s.", vault.name)
    raise
```

- For API details, see [DeleteVault](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an archive

The following code example shows how to delete an Amazon S3 Glacier archive.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    @staticmethod
    def delete_archive(archive):
        """
        Deletes an archive from a vault.

        :param archive: The archive to delete.
        """
        try:
            archive.delete()
            logger.info(
                "Deleted archive %s from vault %s.", archive.id, archive.vault_name)
        except ClientError:
            logger.exception("Couldn't delete archive %s.", archive.id)
            raise
```

- For API details, see [DeleteArchive](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete vault notifications

The following code example shows how to delete Amazon S3 Glacier vault notifications.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    @staticmethod
    def stop_notifications(notification):
        """
        Stops notifications to the configured Amazon SNS topic.

        :param notification: The notification configuration to remove.
        """
        try:
            notification.delete()
            logger.info("Notifications stopped.")
        except ClientError:
            logger.exception("Couldn't stop notifications.")
            raise
```

- For API details, see [DeleteVaultNotifications](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a job

The following code example shows how to describe an Amazon S3 Glacier job.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    @staticmethod
    def get_job_status(job):
        """
        Gets the status of a job.

        :param job: The job to query.
        :return: The current status of the job.
        """
        try:
            job.load()
            logger.info(
                "Job %s is performing action %s and has status %s.", job.id,
                job.action, job.status_code)
        except ClientError:
            logger.exception("Couldn't get status for job %s.", job.id)
            raise
```

```
        else:
            return job.status_code
```

- For API details, see [DescribeJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get job output

The following code example shows how to get Amazon S3 Glacier job output.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    @staticmethod
    def get_job_output(job):
        """
        Gets the output of a job, such as a vault inventory or the contents of an
        archive.

        :param job: The job to get output from.
        :return: The job output, in bytes.
        """
        try:
            response = job.get_output()
            out_bytes = response['body'].read()
            logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)
            if 'archiveDescription' in response:
                logger.info(
                    "These bytes are described as '%s'",
                    response['archiveDescription'])
            except ClientError:
                logger.exception("Couldn't get output for job %s.", job.id)
                raise
        else:
            return out_bytes
```

- For API details, see [GetJobOutput](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get vault notification configuration

The following code example shows how to get Amazon S3 Glacier vault notification configuration.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    @staticmethod
    def get_notification(vault):
        """
        Gets the currently notification configuration for a vault.

        :param vault: The vault to query.
        :return: The notification configuration for the specified vault.
        """
        try:
            notification = vault.Notification()
            logger.info(
                "Vault %s notifies %s on %s events.", vault.name,
                notification.sns_topic, notification.events)
        except ClientError:
            logger.exception("Couldn't get notification data for %s.", vault.name)
            raise
        else:
            return notification
```

- For API details, see [GetVaultNotifications](#) in *AWS SDK for Python (Boto3) API Reference*.

## List jobs

The following code example shows how to list Amazon S3 Glacier jobs.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        :return: The list of jobs of the requested type.
        """
        job_list = []
        try:
```

```
if job_type == 'all':
    jobs = vault.jobs.all()
elif job_type == 'in_progress':
    jobs = vault.jobs_in_progress.all()
elif job_type == 'completed':
    jobs = vault.completed_jobs.all()
elif job_type == 'succeeded':
    jobs = vault.succeeded_jobs.all()
elif job_type == 'failed':
    jobs = vault.failed_jobs.all()
else:
    jobs = []
    logger.warning("%s isn't a type of job I can get.", job_type)
for job in jobs:
    job_list.append(job)
    logger.info("Got %s %s job %s.", job_type, job.action, job.id)
except ClientError:
    logger.exception("Couldn't get %s jobs from %s.", job_type, vault.name)
    raise
else:
    return job_list
```

- For API details, see [ListJobs](#) in *AWS SDK for Python (Boto3) API Reference*.

## List vaults

The following code example shows how to list Amazon S3 Glacier vaults.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    def list_vaults(self):
        """
        Lists vaults for the current account.
        """
        try:
            for vault in self.glacier_resource.vaults.all():
                logger.info("Got vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise
```

- For API details, see [ListVaults](#) in *AWS SDK for Python (Boto3) API Reference*.

## Retrieve a vault inventory

The following code example shows how to retrieve an Amazon S3 Glacier vault inventory.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:  
    """Encapsulates Amazon S3 Glacier API operations."""  
    def __init__(self, glacier_resource):  
        """  
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.  
        """  
        self.glacier_resource = glacier_resource  
  
    @staticmethod  
    def initiate_inventory_retrieval(vault):  
        """  
        Initiates an inventory retrieval job. The inventory describes the contents  
        of the vault. Standard retrievals typically complete within 3–5 hours.  
        When the job completes, you can get the inventory by calling get_output().  
  
        :param vault: The vault to inventory.  
        :return: The inventory retrieval job.  
        """  
        try:  
            job = vault.initiate_inventory_retrieval()  
            logger.info("Started %s job with ID %s.", job.action, job.id)  
        except ClientError:  
            logger.exception("Couldn't start job on vault %s.", vault.name)  
            raise  
        else:  
            return job
```

- For API details, see [InitiateJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Retrieve an archive from a vault

The following code example shows how to retrieve an archive from an Amazon S3 Glacier vault.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:  
    """Encapsulates Amazon S3 Glacier API operations."""  
    def __init__(self, glacier_resource):  
        """  
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.  
        """  
        self.glacier_resource = glacier_resource  
  
    @staticmethod  
    def initiate_archive_retrieval(archive):  
        """  
        Initiates an archive retrieval job. Standard retrievals typically complete  
        within 3–5 hours. When the job completes, you can get the archive contents
```

```
    by calling get_output().  
  
    :param archive: The archive to retrieve.  
    :return: The archive retrieval job.  
    """  
    try:  
        job = archive.initiate_archive_retrieval()  
        logger.info("Started %s job with ID %s.", job.action, job.id)  
    except ClientError:  
        logger.exception("Couldn't start job on archive %s.", archive.id)  
    raise  
else:  
    return job
```

- For API details, see [InitiateJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Set vault notifications

The following code example shows how to set Amazon S3 Glacier vault notifications.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:  
    """Encapsulates Amazon S3 Glacier API operations."""  
    def __init__(self, glacier_resource):  
        """  
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.  
        """  
        self.glacier_resource = glacier_resource  
  
    def set_notifications(self, vault, sns_topic_arn):  
        """  
        Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target  
        for notifications. Amazon S3 Glacier publishes messages to this topic for  
        the configured list of events.  
  
        :param vault: The vault to set up to publish notifications.  
        :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that  
            receives notifications.  
        :return: Data about the new notification configuration.  
        """  
        try:  
            notification = self.glacier_resource.Notification('-', vault.name)  
            notification.set(vaultNotificationConfig={  
                'SNSTopic': sns_topic_arn,  
                'Events': ['ArchiveRetrievalCompleted', 'InventoryRetrievalCompleted']  
            })  
            logger.info(  
                "Notifications will be sent to %s for events %s from %s.",  
                notification.sns_topic, notification.events, notification.vault_name)  
        except ClientError:  
            logger.exception(  
                "Couldn't set notifications to %s on %s.", sns_topic_arn, vault.name)  
        raise  
else:  
    return notification
```

- For API details, see [SetVaultNotifications](#) in *AWS SDK for Python (Boto3) API Reference*.

## Upload an archive to a vault

The following code example shows how to upload an archive to an Amazon S3 Glacier vault.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class GlacierWrapper:  
    """Encapsulates Amazon S3 Glacier API operations."""  
    def __init__(self, glacier_resource):  
        """  
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.  
        """  
        self.glacier_resource = glacier_resource  
  
    @staticmethod  
    def upload_archive(vault, archive_description, archive_file):  
        """  
        Uploads an archive to a vault.  
  
        :param vault: The vault where the archive is put.  
        :param archive_description: A description of the archive.  
        :param archive_file: The archive file to put in the vault.  
        :return: The uploaded archive.  
        """  
        try:  
            archive = vault.upload_archive(  
                archiveDescription=archive_description, body=archive_file)  
            logger.info(  
                "Uploaded %s with ID %s to vault %s.", archive_description,  
                archive.id, vault.name)  
        except ClientError:  
            logger.exception(  
                "Couldn't upload %s to %s.", archive_description, vault.name)  
            raise  
        else:  
            return archive
```

- For API details, see [UploadArchive](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Archive a file, get notifications, and initiate a job

The following code example shows how to:

- Create an Amazon S3 Glacier vault.
- Configure the vault to publish notifications to an Amazon Simple Notification Service (Amazon SNS) topic.
- Upload an archive file to the vault.

- Initiate an archive retrieval job.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps S3 Glacier operations.

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    def create_vault(self, vault_name):
        """
        Creates a vault.

        :param vault_name: The name to give the vault.
        :return: The newly created vault.
        """
        try:
            vault = self.glacier_resource.create_vault(vaultName=vault_name)
            logger.info("Created vault %s.", vault_name)
        except ClientError:
            logger.exception("Couldn't create vault %s.", vault_name)
            raise
        else:
            return vault

    def list_vaults(self):
        """
        Lists vaults for the current account.

        """
        try:
            for vault in self.glacier_resource.vaults.all():
                logger.info("Got vault %s.", vault.name)
        except ClientError:
            logger.exception("Couldn't list vaults.")
            raise

    @staticmethod
    def upload_archive(vault, archive_description, archive_file):
        """
        Uploads an archive to a vault.

        :param vault: The vault where the archive is put.
        :param archive_description: A description of the archive.
        :param archive_file: The archive file to put in the vault.
        :return: The uploaded archive.
        """


```

```
"""
try:
    archive = vault.upload_archive(
        archiveDescription=archive_description, body=archive_file)
    logger.info(
        "Uploaded %s with ID %s to vault %s.", archive_description,
        archive.id, vault.name)
except ClientError:
    logger.exception(
        "Couldn't upload %s to %s.", archive_description, vault.name)
    raise
else:
    return archive

@staticmethod
def initiate_archive_retrieval(archive):
    """
    Initiates an archive retrieval job. Standard retrievals typically complete
    within 3-5 hours. When the job completes, you can get the archive contents
    by calling get_output().

    :param archive: The archive to retrieve.
    :return: The archive retrieval job.
    """
    try:
        job = archive.initiate_archive_retrieval()
        logger.info("Started %s job with ID %s.", job.action, job.id)
    except ClientError:
        logger.exception("Couldn't start job on archive %s.", archive.id)
        raise
    else:
        return job

@staticmethod
def list_jobs(vault, job_type):
    """
    Lists jobs by type for the specified vault.

    :param vault: The vault to query.
    :param job_type: The type of job to list.
    :return: The list of jobs of the requested type.
    """
    job_list = []
    try:
        if job_type == 'all':
            jobs = vault.jobs.all()
        elif job_type == 'in_progress':
            jobs = vault.jobs_in_progress.all()
        elif job_type == 'completed':
            jobs = vault.completed_jobs.all()
        elif job_type == 'succeeded':
            jobs = vault.succeeded_jobs.all()
        elif job_type == 'failed':
            jobs = vault.failed_jobs.all()
        else:
            jobs = []
            logger.warning("%s isn't a type of job I can get.", job_type)
        for job in jobs:
            job_list.append(job)
            logger.info("Got %s %s job %s.", job_type, job.action, job.id)
    except ClientError:
        logger.exception("Couldn't get %s jobs from %s.", job_type, vault.name)
        raise
    else:
        return job_list
```

```
def set_notifications(self, vault, sns_topic_arn):
    """
    Sets an Amazon Simple Notification Service (Amazon SNS) topic as a target
    for notifications. Amazon S3 Glacier publishes messages to this topic for
    the configured list of events.

    :param vault: The vault to set up to publish notifications.
    :param sns_topic_arn: The Amazon Resource Name (ARN) of the topic that
                         receives notifications.
    :return: Data about the new notification configuration.
    """
    try:
        notification = self.glacier_resource.Notification('-', vault.name)
        notification.set(vaultNotificationConfig={
            'SNSTopic': sns_topic_arn,
            'Events': ['ArchiveRetrievalCompleted', 'InventoryRetrievalCompleted']
        })
        logger.info(
            "Notifications will be sent to %s for events %s from %s.",
            notification.sns_topic, notification.events, notification.vault_name)
    except ClientError:
        logger.exception(
            "Couldn't set notifications to %s on %s.", sns_topic_arn, vault.name)
        raise
    else:
        return notification
```

Call functions on the wrapper class to create a vault and upload a file, then configure the vault to publish notifications and initiate a job to retrieve the archive.

```
def upload_demo(glacier, vault_name, topic_arn):
    """
    Shows how to:
    * Create a vault.
    * Configure the vault to publish notifications to an Amazon SNS topic.
    * Upload an archive.
    * Start a job to retrieve the archive.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to create.
    :param topic_arn: The ARN of an Amazon SNS topic that receives notification of
                     Amazon S3 Glacier events.
    """
    print(f"\nCreating vault {vault_name}.")
    vault = glacier.create_vault(vault_name)
    print("\nList of vaults in your account:")
    glacier.list_vaults()
    print(f"\nUploading glacier_basics.py to {vault.name}.")
    with open("glacier_basics.py", 'rb') as upload_file:
        archive = glacier.upload_archive(vault, "glacier_basics.py", upload_file)
    print("\nStarting an archive retrieval request to get the file back from the "
         "vault.")
    glacier.initiate_archive_retrieval(archive)
    print("\nListing in progress jobs:")
    glacier.list_jobs(vault, 'in_progress')
    print("\nBecause Amazon S3 Glacier is intended for infrequent retrieval, an "
          "archive request with Standard retrieval typically completes within 3-5 "
          "hours.")
    if topic_arn:
        notification = glacier.set_notifications(vault, topic_arn)
        print(f"\nVault {vault.name} is configured to notify the "
              f"{notification.sns_topic} topic when {notification.events} "
              "events occur. You can subscribe to this topic to receive "
```

```
f"a message when the archive retrieval completes.\n")
else:
    print(f"\nVault {vault.name} is not configured to notify an Amazon SNS topic "
          f"when the archive retrieval completes so wait a few hours.")
    print("\nRetrieve your job output by running this script with the --retrieve
flag.")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateVault](#)
  - [InitiateJob](#)
  - [ListJobs](#)
  - [ListVaults](#)
  - [SetVaultNotifications](#)
  - [UploadArchive](#)

## Get archive content and delete the archive

The following code example shows how to:

- List jobs for an Amazon S3 Glacier vault and get job status.
- Get the output of a completed archive retrieval job.
- Delete an archive.
- Delete a vault.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps S3 Glacier operations.

```
import argparse
import logging
import os
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

class GlacierWrapper:
    """Encapsulates Amazon S3 Glacier API operations."""
    def __init__(self, glacier_resource):
        """
        :param glacier_resource: A Boto3 Amazon S3 Glacier resource.
        """
        self.glacier_resource = glacier_resource

    @staticmethod
    def list_jobs(vault, job_type):
        """
        Lists jobs by type for the specified vault.

        :param vault: The vault to query.
        :param job_type: The type of job to list.
        """
```

```
:return: The list of jobs of the requested type.  
"""  
job_list = []  
try:  
    if job_type == 'all':  
        jobs = vault.jobs.all()  
    elif job_type == 'in_progress':  
        jobs = vault.jobs_in_progress.all()  
    elif job_type == 'completed':  
        jobs = vault.completed_jobs.all()  
    elif job_type == 'succeeded':  
        jobs = vault.succeeded_jobs.all()  
    elif job_type == 'failed':  
        jobs = vault.failed_jobs.all()  
    else:  
        jobs = []  
        logger.warning("%s isn't a type of job I can get.", job_type)  
    for job in jobs:  
        job_list.append(job)  
        logger.info("Got %s %s job %s.", job_type, job.action, job.id)  
except ClientError:  
    logger.exception("Couldn't get %s jobs from %s.", job_type, vault.name)  
    raise  
else:  
    return job_list  
  
@staticmethod  
def get_job_output(job):  
    """  
    Gets the output of a job, such as a vault inventory or the contents of an  
    archive.  
  
    :param job: The job to get output from.  
    :return: The job output, in bytes.  
    """  
    try:  
        response = job.get_output()  
        out_bytes = response['body'].read()  
        logger.info("Read %s bytes from job %s.", len(out_bytes), job.id)  
        if 'archiveDescription' in response:  
            logger.info(  
                "These bytes are described as '%s'",  
            response['archiveDescription'])  
        except ClientError:  
            logger.exception("Couldn't get output for job %s.", job.id)  
            raise  
        else:  
            return out_bytes  
  
    @staticmethod  
    def delete_archive(archive):  
        """  
        Deletes an archive from a vault.  
  
        :param archive: The archive to delete.  
        """  
        try:  
            archive.delete()  
            logger.info(  
                "Deleted archive %s from vault %s.", archive.id, archive.vault_name)  
        except ClientError:  
            logger.exception("Couldn't delete archive %s.", archive.id)  
            raise  
  
    @staticmethod  
    def delete_vault(vault):
```

```
"""
Deletes a vault.

:param vault: The vault to delete.
"""
try:
    vault.delete()
    logger.info("Deleted vault %s.", vault.name)
except ClientError:
    logger.exception("Couldn't delete vault %s.", vault.name)
    raise
```

Call functions on the wrapper class to get archive content from a completed job, then delete the archive.

```
def retrieve_demo(glacier, vault_name):
    """
    Shows how to:
    * List jobs for a vault and get job status.
    * Get the output of a completed archive retrieval job.
    * Delete an archive.
    * Delete a vault.

    :param glacier: A Boto3 Amazon S3 Glacier resource.
    :param vault_name: The name of the vault to query for jobs.
    """
    vault = glacier.glacier_resource.Vault('-', vault_name)
    try:
        vault.load()
    except ClientError as err:
        if err.response['Error']['Code'] == 'ResourceNotFoundException':
            print(f"\nVault {vault_name} doesn't exist. You must first run this script"
                 f"with the --upload flag to create the vault.")
            return
        else:
            raise
    print(f"\nGetting completed jobs for {vault.name}.")
    jobs = glacier.list_jobs(vault, 'completed')
    if not jobs:
        print("\nNo completed jobs found. Give it some time and try again later.")
        return

    retrieval_job = None
    for job in jobs:
        if job.action == 'ArchiveRetrieval' and job.status_code == 'Succeeded':
            retrieval_job = job
            break
    if retrieval_job is None:
        print("\nNo ArchiveRetrieval jobs found. Give it some time and try again "
              "later.")
        return

    print(f"\nGetting output from job {retrieval_job.id}.")
    archive_bytes = glacier.get_job_output(retrieval_job)
    archive_str = archive_bytes.decode('utf-8')
    print("\nGot archive data. Printing the first 10 lines.")
    print(os.linesep.join(archive_str.split(os.linesep)[:10]))

    print(f"\nDeleting the archive from {vault.name}.")
    archive = glacier.glacier_resource.Archive(
        '-', vault.name, retrieval_job.archive_id)
```

```
glacier.delete_archive(archive)  
  
print(f"\nDeleting {vault.name}.")  
glacier.delete_vault(vault)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [DeleteArchive](#)
  - [DeleteVault](#)
  - [GetJobOutput](#)
  - [ListJobs](#)

## Amazon SES examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Simple Email Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4217\)](#)
- [Scenarios \(p. 4232\)](#)

## Actions

### Create a receipt filter

The following code example shows how to create an Amazon SES receipt filter that blocks incoming mail from an IP address or range of IP addresses.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesReceiptHandler:  
    """Encapsulates Amazon SES receipt handling functions."""  
    def __init__(self, ses_client, s3_resource):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        :param s3_resource: A Boto3 Amazon S3 resource.  
        """  
        self.ses_client = ses_client  
        self.s3_resource = s3_resource  
  
    def create_receipt_filter(self, filter_name, ip_address_or_range, allow):  
        """  
        Creates a filter that allows or blocks incoming mail from an IP address or  
        range.  
  
        :param filter_name: The name to give the filter.  
        :param ip_address_or_range: The IP address or range to block or allow.  
        :param allow: When True, incoming mail is allowed from the specified IP
```

```
        address or range; otherwise, it is blocked.  
    """  
    try:  
        policy = 'Allow' if allow else 'Block'  
        self.ses_client.create_receipt_filter(  
            Filter={  
                'Name': filter_name,  
                'IpFilter': {  
                    'Cidr': ip_address_or_range,  
                    'Policy': policy}})  
        logger.info(  
            "Created receipt filter %s to %s IP of %s.", filter_name, policy,  
            ip_address_or_range)  
    except ClientError:  
        logger.exception("Couldn't create receipt filter %s.", filter_name)  
        raise
```

- For API details, see [CreateReceiptFilter](#) in *AWS SDK for Python (Boto3) API Reference*.

### Create a receipt rule

The following code example shows how to create an Amazon SES receipt rule.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an Amazon S3 bucket where Amazon SES can put copies of incoming emails and create a rule that copies incoming email to the bucket for a specific list of recipients.

```
class SesReceiptHandler:  
    """Encapsulates Amazon SES receipt handling functions."""  
    def __init__(self, ses_client, s3_resource):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        :param s3_resource: A Boto3 Amazon S3 resource.  
        """  
        self.ses_client = ses_client  
        self.s3_resource = s3_resource  
  
    def create_bucket_for_copy(self, bucket_name):  
        """  
        Creates a bucket that can receive copies of emails from Amazon SES. This  
        includes adding a policy to the bucket that grants Amazon SES permission  
        to put objects in the bucket.  
  
        :param bucket_name: The name of the bucket to create.  
        :return: The newly created bucket.  
        """  
        allow_ses_put_policy = {  
            "Version": "2012-10-17",  
            "Statement": [{  
                "Sid": "AllowSESPut",  
                "Effect": "Allow",  
                "Principal": {  
                    "Service": "ses.amazonaws.com"},  
                "Action": "s3:PutObject",  
                "Resource": f"arn:aws:s3:::{bucket_name}/*"}]}  
        bucket = None  
        try:
```

```
bucket = self.s3_resource.create_bucket(  
    Bucket=bucket_name,  
    CreateBucketConfiguration={  
        'LocationConstraint':  
            self.s3_resource.meta.client.meta.region_name})  
bucket.wait_until_exists()  
bucket.Policy().put(Policy=json.dumps(allow_ses_put_policy))  
logger.info("Created bucket %s to receive copies of emails.", bucket_name)  
except ClientError:  
    logger.exception("Couldn't create bucket to receive copies of emails.")  
    if bucket is not None:  
        bucket.delete()  
    raise  
else:  
    return bucket  
  
def create_s3_copy_rule(  
    self, rule_set_name, rule_name, recipients, bucket_name, prefix):  
    """  
    Creates a rule so that all emails received by the specified recipients are  
    copied to an Amazon S3 bucket.  
  
    :param rule_set_name: The name of a previously created rule set to contain  
        this rule.  
    :param rule_name: The name to give the rule.  
    :param recipients: When an email is received by one of these recipients, it  
        is copied to the Amazon S3 bucket.  
    :param bucket_name: The name of the bucket to receive email copies. This  
        bucket must allow Amazon SES to put objects into it.  
    :param prefix: An object key prefix to give the emails copied to the bucket.  
    """  
    try:  
        self.ses_client.create_receipt_rule(  
            RuleSetName=rule_set_name,  
            Rule={  
                'Name': rule_name,  
                'Enabled': True,  
                'Recipients': recipients,  
                'Actions': [  
                    {  
                        'S3Action': {  
                            'BucketName': bucket_name,  
                            'ObjectKeyPrefix': prefix  
                        }  
                    }  
                ]  
            })  
        logger.info(  
            "Created rule %s to copy mail received by %s to bucket %s.",  
            rule_name, recipients, bucket_name)  
    except ClientError:  
        logger.exception("Couldn't create rule %s.", rule_name)  
        raise
```

- For API details, see [CreateReceiptRule](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create a receipt rule set

The following code example shows how to create an Amazon SES receipt rule set to organize rules applied to incoming emails.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesReceiptHandler:  
    """Encapsulates Amazon SES receipt handling functions."""  
    def __init__(self, ses_client, s3_resource):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        :param s3_resource: A Boto3 Amazon S3 resource.  
        """  
        self.ses_client = ses_client  
        self.s3_resource = s3_resource  
  
    def create_receipt_rule_set(self, rule_set_name):  
        """  
        Creates an empty rule set. Rule sets contain individual rules and can be  
        used to organize rules.  
  
        :param rule_set_name: The name to give the rule set.  
        """  
        try:  
            self.ses_client.create_receipt_rule_set(RuleSetName=rule_set_name)  
            logger.info("Created receipt rule set %s.", rule_set_name)  
        except ClientError:  
            logger.exception("Couldn't create receipt rule set %s.", rule_set_name)  
            raise
```

- For API details, see [CreateReceiptRuleSet](#) in *AWS SDK for Python (Boto3) API Reference*.

## Create an email template

The following code example shows how to create an Amazon SES email template.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesTemplate:  
    """Encapsulates Amazon SES template functions."""  
    def __init__(self, ses_client):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        """  
        self.ses_client = ses_client  
        self.template = None  
        self.template_tags = set()  
  
    def _extract_tags(self, subject, text, html):  
        """  
        Extracts tags from a template as a set of unique values.  
  
        :param subject: The subject of the email.  
        :param text: The text version of the email.  
        :param html: The html version of the email.  
        """  
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))  
        logger.info("Extracted template tags: %s", self.template_tags)  
  
    def create_template(self, name, subject, text, html):  
        """  
        Creates an email template.
```

```
:param name: The name of the template.
:param subject: The subject of the email.
:param text: The plain text version of the email.
:param html: The HTML version of the email.
"""
try:
    template = {
        'TemplateName': name,
        'SubjectPart': subject,
        'TextPart': text,
        'HtmlPart': html}
    self.ses_client.create_template(Template=template)
    logger.info("Created template %s.", name)
    self.template = template
    self._extract_tags(subject, text, html)
except ClientError:
    logger.exception("Couldn't create template %s.", name)
    raise
```

- For API details, see [CreateTemplate in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete a receipt filter

The following code example shows how to delete an Amazon SES receipt filter.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""
    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource

    def delete_receipt_filter(self, filter_name):
        """
        Deletes a receipt filter.

        :param filter_name: The name of the filter to delete.
        """
        try:
            self.ses_client.delete_receipt_filter(FilterName=filter_name)
            logger.info("Deleted receipt filter %s.", filter_name)
        except ClientError:
            logger.exception("Couldn't delete receipt filter %s.", filter_name)
            raise
```

- For API details, see [DeleteReceiptFilter in AWS SDK for Python \(Boto3\) API Reference](#).

## Delete a receipt rule

The following code example shows how to delete an Amazon SES receipt rule.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesReceiptHandler:  
    """Encapsulates Amazon SES receipt handling functions."""  
    def __init__(self, ses_client, s3_resource):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        :param s3_resource: A Boto3 Amazon S3 resource.  
        """  
        self.ses_client = ses_client  
        self.s3_resource = s3_resource  
  
    def delete_receipt_rule(self, rule_set_name, rule_name):  
        """  
        Deletes a rule.  
  
        :param rule_set_name: The rule set that contains the rule to delete.  
        :param rule_name: The rule to delete.  
        """  
        try:  
            self.ses_client.delete_receipt_rule(  
                RuleSetName=rule_set_name, RuleName=rule_name)  
            logger.info("Removed rule %s from rule set %s.", rule_name, rule_set_name)  
        except ClientError:  
            logger.exception(  
                "Couldn't remove rule %s from rule set %s.", rule_name, rule_set_name)  
            raise
```

- For API details, see [DeleteReceiptRule](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a rule set

The following code example shows how to delete an Amazon SES rule set and all of the rules it contains.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesReceiptHandler:  
    """Encapsulates Amazon SES receipt handling functions."""  
    def __init__(self, ses_client, s3_resource):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        :param s3_resource: A Boto3 Amazon S3 resource.  
        """  
        self.ses_client = ses_client  
        self.s3_resource = s3_resource  
  
    def delete_receipt_rule_set(self, rule_set_name):  
        """  
        Deletes a rule set. When a rule set is deleted, all of the rules it contains  
        are also deleted.  
  
        :param rule_set_name: The name of the rule set to delete.  
        """
```

```
try:
    self.ses_client.delete_receipt_rule_set(RuleSetName=rule_set_name)
    logger.info("Deleted rule set %s.", rule_set_name)
except ClientError:
    logger.exception("Couldn't delete rule set %s.", rule_set_name)
    raise
```

- For API details, see [DeleteReceiptRuleSet](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an email template

The following code example shows how to delete an Amazon SES email template.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)

    def delete_template(self):
        """
        Deletes an email template.
        """
        try:
            self.ses_client.delete_template(TemplateName=self.template['TemplateName'])
            logger.info("Deleted template %s.", self.template['TemplateName'])
            self.template = None
            self.template_tags = None
        except ClientError:
            logger.exception(
                "Couldn't delete template %s.", self.template['TemplateName'])
            raise
```

- For API details, see [DeleteTemplate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete an identity

The following code example shows how to delete an Amazon SES identity.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesIdentity:  
    """Encapsulates Amazon SES identity functions."""  
    def __init__(self, ses_client):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        """  
        self.ses_client = ses_client  
  
    def delete_identity(self, identity):  
        """  
        Deletes an identity.  
  
        :param identity: The identity to remove.  
        """  
        try:  
            self.ses_client.delete_identity(Identity=identity)  
            logger.info("Deleted identity %s.", identity)  
        except ClientError:  
            logger.exception("Couldn't delete identity %s.", identity)  
            raise
```

- For API details, see [DeleteIdentity](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a receipt rule set

The following code example shows how to describe an Amazon SES receipt rule set.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesReceiptHandler:  
    """Encapsulates Amazon SES receipt handling functions."""  
    def __init__(self, ses_client, s3_resource):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        :param s3_resource: A Boto3 Amazon S3 resource.  
        """  
        self.ses_client = ses_client  
        self.s3_resource = s3_resource  
  
    def describe_receipt_rule_set(self, rule_set_name):  
        """  
        Gets data about a rule set.  
  
        :param rule_set_name: The name of the rule set to retrieve.  
        :return: Data about the rule set.  
        """  
        try:  
            response = self.ses_client.describe_receipt_rule_set(  
                RuleSetName=rule_set_name)  
            logger.info("Got data for rule set %s.", rule_set_name)
```

```
        except ClientError:
            logger.exception("Couldn't get data for rule set %s.", rule_set_name)
            raise
        else:
            return response
```

- For API details, see [DescribeReceiptRuleSet](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get an existing email template

The following code example shows how to get an existing Amazon SES email template.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
        """
        Extracts tags from a template as a set of unique values.

        :param subject: The subject of the email.
        :param text: The text version of the email.
        :param html: The html version of the email.
        """
        self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
        logger.info("Extracted template tags: %s", self.template_tags)

    def get_template(self, name):
        """
        Gets a previously created email template.

        :param name: The name of the template to retrieve.
        :return: The retrieved email template.
        """
        try:
            response = self.ses_client.get_template(TemplateName=name)
            self.template = response['Template']
            logger.info("Got template %s.", name)
            self._extract_tags(
                self.template['SubjectPart'], self.template['TextPart'],
                self.template['HtmlPart'])
        except ClientError:
            logger.exception("Couldn't get template %s.", name)
            raise
        else:
            return self.template
```

- For API details, see [GetTemplate](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get the status of an identity

The following code example shows how to get the status of an Amazon SES identity.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client

    def get_identity_status(self, identity):
        """
        Gets the status of an identity. This can be used to discover whether
        an identity has been successfully verified.

        :param identity: The identity to query.
        :return: The status of the identity.
        """
        try:
            response = self.ses_client.get_identity_verification_attributes(
                Identities=[identity])
            status = response['VerificationAttributes'].get(
                identity, {'VerificationStatus': 'NotFound'})['VerificationStatus']
            logger.info("Got status of %s for %s.", status, identity)
        except ClientError:
            logger.exception("Couldn't get status for %s.", identity)
            raise
        else:
            return status
```

- For API details, see [GetIdentityVerificationAttributes](#) in *AWS SDK for Python (Boto3) API Reference*.

## List email templates

The following code example shows how to list Amazon SES email templates.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()
```

```
def _extract_tags(self, subject, text, html):
    """
    Extracts tags from a template as a set of unique values.

    :param subject: The subject of the email.
    :param text: The text version of the email.
    :param html: The html version of the email.
    """
    self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
    logger.info("Extracted template tags: %s", self.template_tags)

def list_templates(self):
    """
    Gets a list of all email templates for the current account.

    :return: The list of retrieved email templates.
    """
    try:
        response = self.ses_client.list_templates()
        templates = response['TemplatesMetadata']
        logger.info("Got %s templates.", len(templates))
    except ClientError:
        logger.exception("Couldn't get templates.")
        raise
    else:
        return templates
```

- For API details, see [ListTemplates](#) in *AWS SDK for Python (Boto3) API Reference*.

## List identities

The following code example shows how to list Amazon SES identities.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesIdentity:
    """
    Encapsulates Amazon SES identity functions.
    """
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client

    def list_identities(self, identity_type, max_items):
        """
        Gets the identities of the specified type for the current account.

        :param identity_type: The type of identity to retrieve, such as EmailAddress.
        :param max_items: The maximum number of identities to retrieve.
        :return: The list of retrieved identities.
        """
        try:
            response = self.ses_client.list_identities(
                IdentityType=identity_type, MaxItems=max_items)
            identities = response['Identities']
            logger.info("Got %s identities for the current account.", len(identities))
        except ClientError:
            logger.exception("Couldn't list identities for the current account.")
```

```
        raise
else:
    return identities
```

- For API details, see [ListIdentities](#) in *AWS SDK for Python (Boto3) API Reference*.

## List receipt filters

The following code example shows how to list Amazon SES receipt filters.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesReceiptHandler:
    """Encapsulates Amazon SES receipt handling functions."""
    def __init__(self, ses_client, s3_resource):
        """
        :param ses_client: A Boto3 Amazon SES client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        """
        self.ses_client = ses_client
        self.s3_resource = s3_resource

    def list_receipt_filters(self):
        """
        Gets the list of receipt filters for the current account.

        :return: The list of receipt filters.
        """
        try:
            response = self.ses_client.list_receipt_filters()
            filters = response['Filters']
            logger.info("Got %s receipt filters.", len(filters))
        except ClientError:
            logger.exception("Couldn't get receipt filters.")
            raise
        else:
            return filters
```

- For API details, see [ListReceiptFilters](#) in *AWS SDK for Python (Boto3) API Reference*.

## Send email

The following code example shows how to send email with Amazon SES.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""
    def __init__(self, ses_client):
```

```
"""
:param ses_client: A Boto3 Amazon SES client.
"""
self.ses_client = ses_client

def send_email(self, source, destination, subject, text, html, reply_tos=None):
    """
    Sends an email.

    Note: If your account is in the Amazon SES sandbox, the source and
    destination email accounts must both be verified.

    :param source: The source email account.
    :param destination: The destination email account.
    :param subject: The subject of the email.
    :param text: The plain text version of the body of the email.
    :param html: The HTML version of the body of the email.
    :param reply_tos: Email accounts that will receive a reply if the recipient
                      replies to the message.
    :return: The ID of the message, assigned by Amazon SES.
"""

send_args = {
    'Source': source,
    'Destination': destination.to_service_format(),
    'Message': {
        'Subject': {'Data': subject},
        'Body': {'Text': {'Data': text}, 'Html': {'Data': html}}}
if reply_tos is not None:
    send_args['ReplyToAddresses'] = reply_tos
try:
    response = self.ses_client.send_email(**send_args)
    message_id = response['MessageId']
    logger.info(
        "Sent mail %s from %s to %s.", message_id, source, destination.tos)
except ClientError:
    logger.exception(
        "Couldn't send mail from %s to %s.", source, destination.tos)
    raise
else:
    return message_id
```

- For API details, see [SendEmail](#) in *AWS SDK for Python (Boto3) API Reference*.

## Send templated email

The following code example shows how to send templated email with Amazon SES.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client

    def send_templated_email(
```

```
        self, source, destination, template_name, template_data,
        reply_tos=None):
"""
Sends an email based on a template. A template contains replaceable tags
each enclosed in two curly braces, such as {{name}}. The template data passed
in this function contains key-value pairs that define the values to insert
in place of the template tags.

Note: If your account is in the Amazon SES sandbox, the source and
destination email accounts must both be verified.

:param source: The source email account.
:param destination: The destination email account.
:param template_name: The name of a previously created template.
:param template_data: JSON-formatted key-value pairs of replacement values
    that are inserted in the template before it is sent.
:return: The ID of the message, assigned by Amazon SES.
"""

send_args = {
    'Source': source,
    'Destination': destination.to_service_format(),
    'Template': template_name,
    'TemplateData': json.dumps(template_data)
}
if reply_tos is not None:
    send_args['ReplyToAddresses'] = reply_tos
try:
    response = self.ses_client.send templated_email(**send_args)
    message_id = response['MessageId']
    logger.info(
        "Sent templated mail %s from %s to %s.", message_id, source,
        destination.tos)
except ClientError:
    logger.exception(
        "Couldn't send templated mail from %s to %s.", source, destination.tos)
    raise
else:
    return message_id
```

- For API details, see [SendTemplatedEmail](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update an email template

The following code example shows how to update an Amazon SES email template.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesTemplate:
    """Encapsulates Amazon SES template functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client
        self.template = None
        self.template_tags = set()

    def _extract_tags(self, subject, text, html):
```

```
"""
Extracts tags from a template as a set of unique values.

:param subject: The subject of the email.
:param text: The text version of the email.
:param html: The html version of the email.
"""

self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
logger.info("Extracted template tags: %s", self.template_tags)

def update_template(self, name, subject, text, html):
    """
    Updates a previously created email template.

    :param name: The name of the template.
    :param subject: The subject of the email.
    :param text: The plain text version of the email.
    :param html: The HTML version of the email.
    """

    try:
        template = {
            'TemplateName': name,
            'SubjectPart': subject,
            'TextPart': text,
            'HtmlPart': html}
        self.ses_client.update_template(Template=template)
        logger.info("Updated template %s.", name)
        self.template = template
        self._extract_tags(subject, text, html)
    except ClientError:
        logger.exception("Couldn't update template %s.", name)
        raise
```

- For API details, see [UpdateTemplate in AWS SDK for Python \(Boto3\) API Reference](#).

## Verify a domain identity

The following code example shows how to verify a domain identity with Amazon SES.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesIdentity:
    """
    Encapsulates Amazon SES identity functions.
    """
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client

    def verify_domain_identity(self, domain_name):
        """
        Starts verification of a domain identity. To complete verification, you must
        create a TXT record with a specific format through your DNS provider.

        For more information, see *Verifying a domain with Amazon SES* in the
        Amazon SES documentation:
            https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domain-
        procedure.html
        """
```

```
:param domain_name: The name of the domain to verify.  
:return: The token to include in the TXT record with your DNS provider.  
"""  
try:  
    response = self.ses_client.verify_domain_identity(Domain=domain_name)  
    token = response['VerificationToken']  
    logger.info("Got domain verification token for %s.", domain_name)  
except ClientError:  
    logger.exception("Couldn't verify domain %s.", domain_name)  
    raise  
else:  
    return token
```

- For API details, see [VerifyDomainIdentity](#) in *AWS SDK for Python (Boto3) API Reference*.

## Verify an email identity

The following code example shows how to verify an email identity with Amazon SES.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SesIdentity:  
    """Encapsulates Amazon SES identity functions."""  
    def __init__(self, ses_client):  
        """  
        :param ses_client: A Boto3 Amazon SES client.  
        """  
        self.ses_client = ses_client  
  
    def verify_email_identity(self, email_address):  
        """  
        Starts verification of an email identity. This function causes an email  
        to be sent to the specified email address from Amazon SES. To complete  
        verification, follow the instructions in the email.  
  
        :param email_address: The email address to verify.  
        """  
        try:  
            self.ses_client.verify_email_identity(EmailAddress=email_address)  
            logger.info("Started verification of %s.", email_address)  
        except ClientError:  
            logger.exception("Couldn't start verification of %s.", email_address)  
            raise
```

- For API details, see [VerifyEmailIdentity](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Copy email and domain identities from one AWS Region to another

The following code example shows how to copy Amazon SES email and domain identities from one AWS Region to another. When domain identities are managed by Route 53, verification records are copied to the domain for the destination Region.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import argparse
import json
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def get_identities(ses_client):
    """
    Gets the identities for the current Region. The Region is specified in the
    Boto3 Amazon SES client object.

    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of email identities and the list of domain identities.
    """
    email_identities = []
    domain_identities = []
    try:
        identityPaginator = ses_client.getPaginator('list_identities')
        identityIterator = identityPaginator.paginate(
            PaginationConfig={'PageSize': 20})
        for identityPage in identityIterator:
            for identity in identityPage['Identities']:
                if '@' in identity:
                    email_identities.append(identity)
                else:
                    domain_identities.append(identity)
        logger.info(
            "Found %s email and %s domain identities.", len(email_identities),
            len(domain_identities))
    except ClientError:
        logger.exception("Couldn't get identities.")
        raise
    else:
        return email_identities, domain_identities

def verify_emails(email_list, ses_client):
    """
    Starts verification of a list of email addresses. Verification causes an email
    to be sent to each address. To complete verification, the recipient must follow
    the instructions in the email.

    :param email_list: The list of email addresses to verify.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of emails that were successfully submitted for verification.
    """
    verified_emails = []
    for email in email_list:
        try:
            ses_client.verify_email_identity(EmailAddress=email)
            verified_emails.append(email)
            logger.info("Started verification of %s.", email)
        except ClientError:
            logger.warning("Couldn't start verification of %s.", email)
```

```
    return verified_emails

def verify_domains(domain_list, ses_client):
    """
    Starts verification for a list of domain identities. This returns a token for
    each domain, which must be registered as a TXT record with the DNS provider for
    the domain.

    :param domain_list: The list of domains to verify.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The generated domain tokens to use to completed verification.
    """
    domain_tokens = []
    for domain in domain_list:
        try:
            response = ses_client.verify_domain_identity(Domain=domain)
            token = response['VerificationToken']
            domain_tokens[domain] = token
            logger.info("Got verification token %s for domain %s.", token, domain)
        except ClientError:
            logger.warning("Couldn't get verification token for domain %s.", domain)
    return domain_tokens

def get_hosted_zones(route53_client):
    """
    Gets the Amazon Route 53 hosted zones for the current account.

    :param route53_client: A Boto3 Route 53 client.
    :return: The list of hosted zones.
    """
    zones = []
    try:
        zonePaginator = route53_client.get_paginator('list_hosted_zones')
        zoneIterator = zonePaginator.paginate(PaginationConfig={'PageSize': 20})
        zones = [
            zone for zonePage in zoneIterator for zone in zonePage['HostedZones']]
        logger.info("Found %s hosted zones.", len(zones))
    except ClientError:
        logger.warning("Couldn't get hosted zones.")
    return zones

def find_domain_zone_matches(domains, zones):
    """
    Finds matches between Amazon SES verified domains and Route 53 hosted zones.
    Subdomain matches are taken when found, otherwise root domain matches are taken.

    :param domains: The list of domains to match.
    :param zones: The list of hosted zones to match.
    :return: The set of matched domain-zone pairs. When a match is not found, the
            domain is included in the set with a zone value of None.
    """
    domain_zones = {}
    for domain in domains:
        domain_zones[domain] = None
        # Start at the most specific sub-domain and walk up to the root domain until a
        # zone match is found.
        domainSplit = domain.split('.')
        for index in range(0, len(domainSplit) - 1):
            subDomain = '.'.join(domainSplit[index:])
            for zone in zones:
                # Normalize the zone name from Route 53 by removing the trailing '..'.
                zoneName = zone['Name'][:-1]
                if subDomain == zoneName:
```

```

        domain_zones[domain] = zone
        break
    if domain_zones[domain] is not None:
        break
return domain_zones

def add_route53_verification_record(domain, token, zone, route53_client):
    """
    Adds a domain verification TXT record to the specified Route 53 hosted zone.
    When a TXT record already exists in the hosted zone for the specified domain,
    the existing values are preserved and the new token is added to the list.

    :param domain: The domain to add.
    :param token: The verification token for the domain.
    :param zone: The hosted zone where the domain verification record is added.
    :param route53_client: A Boto3 Route 53 client.
    """
    domain_token_record_set_name = f'_amazonses.{domain}'
    record_setPaginator = route53_client.getPaginator(
        'list_resource_record_sets')
    record_setIterator = record_setPaginator.paginate(
        HostedZoneId=zone['Id'], PaginationConfig={'PageSize': 20})
    records = []
    for record_set_page in record_setIterator:
        try:
            txt_record_set = next(
                record_set for record_set
                in record_set_page['ResourceRecordSets']
                if record_set['Name'][:-1] == domain_token_record_set_name and
                record_set['Type'] == 'TXT')
            records = txt_record_set['ResourceRecords']
            logger.info(
                "Existing TXT record found in set %s for zone %s.",
                domain_token_record_set_name, zone['Name'])
            break
        except StopIteration:
            pass
    records.append({'Value': json.dumps(token)})
    changes = [
        {
            'Action': 'UPSERT',
            'ResourceRecordSet': {
                'Name': domain_token_record_set_name,
                'Type': 'TXT',
                'TTL': 1800,
                'ResourceRecords': records}}]
    try:
        route53_client.change_resource_record_sets(
            HostedZoneId=zone['Id'], ChangeBatch={'Changes': changes})
        logger.info(
            "Created or updated the TXT record in set %s for zone %s.",
            domain_token_record_set_name, zone['Name'])
    except ClientError as err:
        logger.warning(
            "Got error %s. Couldn't create or update the TXT record for zone %s.",
            err.response['Error']['Code'], zone['Name'])

def generate_dkim_tokens(domain, ses_client):
    """
    Generates DKIM tokens for a domain. These must be added as CNAME records to the
    DNS provider for the domain.

    :param domain: The domain to generate tokens for.
    :param ses_client: A Boto3 Amazon SES client.
    :return: The list of generated DKIM tokens.
    """

```

```

"""
dkim_tokens = []
try:
    dkim_tokens = ses_client.verify_domain_dkim(Domain=domain)['DkimTokens']
    logger.info("Generated %s DKIM tokens for domain %s.", len(dkim_tokens),
domain)
except ClientError:
    logger.warning("Couldn't generate DKIM tokens for domain %s.", domain)
return dkim_tokens

def add_dkim_domain_tokens(hosted_zone, domain, tokens, route53_client):
"""
    Adds DKIM domain token CNAME records to a Route 53 hosted zone.

:param hosted_zone: The hosted zone where the records are added.
:param domain: The domain to add.
:param tokens: The DKIM tokens for the domain to add.
:param route53_client: A Boto3 Route 53 client.
"""

try:
    changes = [{}
        'Action': 'UPSERT',
        'ResourceRecordSet': {
            'Name': f'{token}._domainkey.{domain}',
            'Type': 'CNAME',
            'TTL': 1800,
            'ResourceRecords': [{'Value': f'{token}.dkim.amazones.com'}]
        } for token in tokens]
    route53_client.change_resource_record_sets(
        HostedZoneId=hosted_zone['Id'], ChangeBatch={'Changes': changes})
    logger.info(
        "Added %s DKIM CNAME records to %s in zone %s.", len(tokens),
        domain, hosted_zone['Name'])
except ClientError:
    logger.warning(
        "Couldn't add DKIM CNAME records for %s to zone %s.", domain,
        hosted_zone['Name'])

def configure_sns_topics(identity, topics, ses_client):
"""
    Configures Amazon Simple Notification Service (Amazon SNS) notifications for
    an identity. The Amazon SNS topics must already exist.

:param identity: The identity to configure.
:param topics: The list of topics to configure. The choices are Bounce, Delivery,
    or Complaint.
:param ses_client: A Boto3 Amazon SES client.
"""

for topic in topics:
    topic_arn = input(
        f"Enter the Amazon Resource Name (ARN) of the {topic} topic or press "
        f"Enter to skip: ")
    if topic_arn != '':
        try:
            ses_client.set_identity_notification_topic(
                Identity=identity, NotificationType=topic, SnsTopic=topic_arn)
            logger.info("Configured %s for %s notifications.", identity, topic)
        except ClientError:
            logger.warning(
                "Couldn't configure %s for %s notifications.", identity, topic)

def replicate(source_client, destination_client, route53_client):
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

```

```

print('*'*88)
print(f'Replicating Amazon SES identities and other configuration from "{source_client.meta.region_name}" to {destination_client.meta.region_name}."')
print('*'*88)

print(f'Retrieving identities from {source_client.meta.region_name}."')
source_emails, source_domains = get_identities(source_client)
print("Email addresses found:")
print(*source_emails)
print("Domains found:")
print(*source_domains)

print("Starting verification for email identities.")
dest_emails = verify_emails(source_emails, destination_client)
print("Getting domain tokens for domain identities.")
dest_domain_tokens = verify_domains(source_domains, destination_client)

# Get Route 53 hosted zones and match them with Amazon SES domains.
answer = input(
    "Is the DNS configuration for your domains managed by Amazon Route 53 (y/n)? ")
use_route53 = answer.lower() == 'y'
hosted_zones = get_hosted_zones(route53_client) if use_route53 else []
if use_route53:
    print("Adding or updating Route 53 TXT records for your domains.")
    domain_zones = find_domain_zone_matches(dest_domain_tokens.keys(),
                                              hosted_zones)
    for domain in domain_zones:
        add_route53_verification_record(
            domain, dest_domain_tokens[domain], domain_zones[domain],
            route53_client)
else:
    print("Use these verification tokens to create TXT records through your DNS "
          "provider:")
    pprint(dest_domain_tokens)

answer = input("Do you want to configure DKIM signing for your identities (y/n)? ")
if answer.lower() == 'y':
    # Build a set of unique domains from email and domain identities.
    domains = {email.split('@')[1] for email in dest_emails}
    domains.update(dest_domain_tokens)
    domain_zones = find_domain_zone_matches(domains, hosted_zones)
    for domain, zone in domain_zones.items():
        answer = input(
            f"Do you want to configure DKIM signing for {domain} (y/n)? ")
        if answer.lower() == 'y':
            dkim_tokens = generate_dkim_tokens(domain, destination_client)
            if use_route53 and zone is not None:
                add_dkim_domain_tokens(zone, domain, dkim_tokens, route53_client)
            else:
                print(
                    "Add the following DKIM tokens as CNAME records through your "
                    "DNS provider:")
                print(*dkim_tokens, sep='\n')

    answer = input(
        "Do you want to configure Amazon SNS notifications for your identities (y/n)? ")
    if answer.lower() == 'y':
        for identity in dest_emails + list(dest_domain_tokens.keys()):
            answer = input(
                f"Do you want to configure Amazon SNS topics for {identity} (y/n)? ")
            if answer.lower() == 'y':
                configure sns_topics(
                    identity, ['Bounce', 'Delivery', 'Complaint'], destination_client)

```

```
print(f'Replication complete for {destination_client.meta.region_name}.")  
print('*'*88)  
  
def main():  
    boto3_session = boto3.Session()  
    ses_regions = boto3_session.get_available_regions('ses')  
    parser = argparse.ArgumentParser(  
        description="Copies email address and domain identities from one AWS Region to  
        another. Optionally adds records for domain verification and DKIM  
        signing to domains that are managed by Amazon Route 53, and sets up Amazon SNS notifications for events of interest.")  
    parser.add_argument(  
        'source_region', choices=ses_regions, help="The region to copy from.")  
    parser.add_argument(  
        'destination_region', choices=ses_regions, help="The region to copy to.")  
    args = parser.parse_args()  
    source_client = boto3.client('ses', region_name=args.source_region)  
    destination_client = boto3.client('ses', region_name=args.destination_region)  
    route53_client = boto3.client('route53')  
    replicate(source_client, destination_client, route53_client)  
  
if __name__ == '__main__':  
    main()
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [ListIdentities](#)
  - [SetIdentityNotificationTopic](#)
  - [VerifyDomainDkim](#)
  - [VerifyDomainIdentity](#)
  - [VerifyEmailIdentity](#)

## Generate credentials to connect to an SMTP endpoint

The following code example shows how to generate credentials to connect to an Amazon SES SMTP endpoint.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#!/usr/bin/env python3  
  
import hmac  
import hashlib  
import base64  
import argparse  
  
SMTP_REGIONS = [  
    'us-east-2',      # US East (Ohio)  
    'us-east-1',      # US East (N. Virginia)  
    'us-west-2',      # US West (Oregon)  
    'ap-south-1',     # Asia Pacific (Mumbai)  
    'ap-northeast-2', # Asia Pacific (Seoul)
```

```

'ap-southeast-1', # Asia Pacific (Singapore)
'ap-southeast-2', # Asia Pacific (Sydney)
'ap-northeast-1', # Asia Pacific (Tokyo)
'ca-central-1', # Canada (Central)
'eu-central-1', # Europe (Frankfurt)
'eu-west-1', # Europe (Ireland)
'eu-west-2', # Europe (London)
'sa-east-1', # South America (Sao Paulo)
'us-gov-west-1', # AWS GovCloud (US)
]

# These values are required to calculate the signature. Do not change them.
DATE = "11111111"
SERVICE = "ses"
MESSAGE = "SendRawEmail"
TERMINAL = "aws4_request"
VERSION = 0x04

def sign(key, msg):
    return hmac.new(key, msg.encode('utf-8'), hashlib.sha256).digest()

def calculate_key(secret_access_key, region):
    if region not in SMTP_REGIONS:
        raise ValueError(f"The {region} Region doesn't have an SMTP endpoint.")

    signature = sign(("AWS4" + secret_access_key).encode('utf-8'), DATE)
    signature = sign(signature, region)
    signature = sign(signature, SERVICE)
    signature = sign(signature, TERMINAL)
    signature = sign(signature, MESSAGE)
    signature_and_version = bytes([VERSION]) + signature
    smtp_password = base64.b64encode(signature_and_version)
    return smtp_password.decode('utf-8')

def main():
    parser = argparse.ArgumentParser(
        description='Convert a Secret Access Key for an IAM user to an SMTP password.')
    parser.add_argument(
        'secret', help='The Secret Access Key to convert.')
    parser.add_argument(
        'region',
        help='The AWS Region where the SMTP password will be used.',
        choices=SMTP_REGIONS)
    args = parser.parse_args()
    print(calculate_key(args.secret, args.region))

if __name__ == '__main__':
    main()

```

## Verify an email identity and send messages

The following code example shows how to:

- Add and verify an email address with Amazon SES.
- Send a standard email message.
- Create a template and send a templated email message.
- Send a message by using an Amazon SES SMTP server.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions to wrap Amazon SES identity actions.

```
class SesIdentity:
    """Encapsulates Amazon SES identity functions."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client

    def verify_domain_identity(self, domain_name):
        """
        Starts verification of a domain identity. To complete verification, you must
        create a TXT record with a specific format through your DNS provider.

        For more information, see *Verifying a domain with Amazon SES* in the
        Amazon SES documentation:
            https://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domain-
procedure.html

        :param domain_name: The name of the domain to verify.
        :return: The token to include in the TXT record with your DNS provider.
        """
        try:
            response = self.ses_client.verify_domain_identity(Domain=domain_name)
            token = response['VerificationToken']
            logger.info("Got domain verification token for %s.", domain_name)
        except ClientError:
            logger.exception("Couldn't verify domain %s.", domain_name)
            raise
        else:
            return token

    def verify_email_identity(self, email_address):
        """
        Starts verification of an email identity. This function causes an email
        to be sent to the specified email address from Amazon SES. To complete
        verification, follow the instructions in the email.

        :param email_address: The email address to verify.
        """
        try:
            self.ses_client.verify_email_identity(EmailAddress=email_address)
            logger.info("Started verification of %s.", email_address)
        except ClientError:
            logger.exception("Couldn't start verification of %s.", email_address)
            raise

    def wait_until_identity_exists(self, identity):
        """
        Waits until an identity exists. The waiter polls Amazon SES until the
        identity has been successfully verified or until it exceeds its maximum time.

        :param identity: The identity to wait for.
        """
        waiter = self.ses_client.get_waiter('identity_exists')
        logger.info("Waiting until %s exists.", identity)
        waiter.wait(Identities=[identity])
```

```

        except WaiterError:
            logger.error("Waiting for identity %s failed or timed out.", identity)
            raise

    def get_identity_status(self, identity):
        """
        Gets the status of an identity. This can be used to discover whether
        an identity has been successfully verified.

        :param identity: The identity to query.
        :return: The status of the identity.
        """
        try:
            response = self.ses_client.get_identity_verification_attributes(
                Identities=[identity])
            status = response['VerificationAttributes'].get(
                identity, {'VerificationStatus': 'NotFound'})['VerificationStatus']
            logger.info("Got status of %s for %s.", status, identity)
        except ClientError:
            logger.exception("Couldn't get status for %s.", identity)
            raise
        else:
            return status

    def delete_identity(self, identity):
        """
        Deletes an identity.

        :param identity: The identity to remove.
        """
        try:
            self.ses_client.delete_identity(Identity=identity)
            logger.info("Deleted identity %s.", identity)
        except ClientError:
            logger.exception("Couldn't delete identity %s.", identity)
            raise

    def list_identities(self, identity_type, max_items):
        """
        Gets the identities of the specified type for the current account.

        :param identity_type: The type of identity to retrieve, such as EmailAddress.
        :param max_items: The maximum number of identities to retrieve.
        :return: The list of retrieved identities.
        """
        try:
            response = self.ses_client.list_identities(
                IdentityType=identity_type, MaxItems=max_items)
            identities = response['Identities']
            logger.info("Got %s identities for the current account.", len(identities))
        except ClientError:
            logger.exception("Couldn't list identities for the current account.")
            raise
        else:
            return identities

```

Create functions to wrap Amazon SES template actions.

```

class SesTemplate:
    """
    Encapsulates Amazon SES template functions.
    """
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """

```

```
self.ses_client = ses_client
self.template = None
self.template_tags = set()

def _extract_tags(self, subject, text, html):
    """
    Extracts tags from a template as a set of unique values.

    :param subject: The subject of the email.
    :param text: The text version of the email.
    :param html: The html version of the email.
    """
    self.template_tags = set(re.findall(TEMPLATE_REGEX, subject + text + html))
    logger.info("Extracted template tags: %s", self.template_tags)

def create_template(self, name, subject, text, html):
    """
    Creates an email template.

    :param name: The name of the template.
    :param subject: The subject of the email.
    :param text: The plain text version of the email.
    :param html: The HTML version of the email.
    """
    try:
        template = {
            'TemplateName': name,
            'SubjectPart': subject,
            'TextPart': text,
            'HtmlPart': html}
        self.ses_client.create_template(Template=template)
        logger.info("Created template %s.", name)
        self.template = template
        self._extract_tags(subject, text, html)
    except ClientError:
        logger.exception("Couldn't create template %s.", name)
        raise

def delete_template(self):
    """
    Deletes an email template.

    """
    try:
        self.ses_client.delete_template(TemplateName=self.template['TemplateName'])
        logger.info("Deleted template %s.", self.template['TemplateName'])
        self.template = None
        self.template_tags = None
    except ClientError:
        logger.exception(
            "Couldn't delete template %s.", self.template['TemplateName'])
        raise

def get_template(self, name):
    """
    Gets a previously created email template.

    :param name: The name of the template to retrieve.
    :return: The retrieved email template.
    """
    try:
        response = self.ses_client.get_template(TemplateName=name)
        self.template = response['Template']
        logger.info("Got template %s.", name)
        self._extract_tags(
            self.template['SubjectPart'], self.template['TextPart'],
            self.template['HtmlPart'])
```

```

        except ClientError:
            logger.exception("Couldn't get template %s.", name)
            raise
        else:
            return self.template

    def list_templates(self):
        """
        Gets a list of all email templates for the current account.

        :return: The list of retrieved email templates.
        """
        try:
            response = self.ses_client.list_templates()
            templates = response['TemplatesMetadata']
            logger.info("Got %s templates.", len(templates))
        except ClientError:
            logger.exception("Couldn't get templates.")
            raise
        else:
            return templates

    def update_template(self, name, subject, text, html):
        """
        Updates a previously created email template.

        :param name: The name of the template.
        :param subject: The subject of the email.
        :param text: The plain text version of the email.
        :param html: The HTML version of the email.
        """
        try:
            template = {
                'TemplateName': name,
                'SubjectPart': subject,
                'TextPart': text,
                'HtmlPart': html}
            self.ses_client.update_template(Template=template)
            logger.info("Updated template %s.", name)
            self.template = template
            self._extract_tags(subject, text, html)
        except ClientError:
            logger.exception("Couldn't update template %s.", name)
            raise

```

Create functions to wrap Amazon SES email actions.

```

class SesDestination:
    """
    Contains data about an email destination.
    """
    def __init__(self, tos=None, ccs=None, bccs=None):
        """
        :param tos: The list of recipients on the 'To:' line.
        :param ccs: The list of recipients on the 'CC:' line.
        :param bccs: The list of recipients on the 'BCC:' line.
        """
        self.tos = tos
        self.ccs = ccs
        self.bccs = bccs

    def to_service_format(self):
        """
        :return: The destination data in the format expected by Amazon SES.
        """
        svc_format = {'ToAddresses': self.tos}

```

```

if self.ccs is not None:
    svc_format['CcAddresses'] = self.ccs
if self.bccs is not None:
    svc_format['BccAddresses'] = self.bccs
return svc_format

class SesMailSender:
    """Encapsulates functions to send emails with Amazon SES."""
    def __init__(self, ses_client):
        """
        :param ses_client: A Boto3 Amazon SES client.
        """
        self.ses_client = ses_client

    def send_email(self, source, destination, subject, text, html, reply_tos=None):
        """
        Sends an email.

        Note: If your account is in the Amazon SES sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param subject: The subject of the email.
        :param text: The plain text version of the body of the email.
        :param html: The HTML version of the body of the email.
        :param reply_tos: Email accounts that will receive a reply if the recipient
            replies to the message.
        :return: The ID of the message, assigned by Amazon SES.
        """
        send_args = {
            'Source': source,
            'Destination': destination.to_service_format(),
            'Message': {
                'Subject': {'Data': subject},
                'Body': {'Text': {'Data': text}, 'Html': {'Data': html}}}}
        if reply_tos is not None:
            send_args['ReplyToAddresses'] = reply_tos
        try:
            response = self.ses_client.send_email(**send_args)
            message_id = response['MessageId']
            logger.info(
                "Sent mail %s from %s to %s.", message_id, source, destination.tos)
        except ClientError:
            logger.exception(
                "Couldn't send mail from %s to %s.", source, destination.tos)
            raise
        else:
            return message_id

    def send_templated_email(
        self, source, destination, template_name, template_data,
        reply_tos=None):
        """
        Sends an email based on a template. A template contains replaceable tags
        each enclosed in two curly braces, such as {{name}}. The template data passed
        in this function contains key-value pairs that define the values to insert
        in place of the template tags.

        Note: If your account is in the Amazon SES sandbox, the source and
        destination email accounts must both be verified.

        :param source: The source email account.
        :param destination: The destination email account.
        :param template_name: The name of a previously created template.
        :param template_data: JSON-formatted key-value pairs of replacement values
        """

```

```
    that are inserted in the template before it is sent.
:return: The ID of the message, assigned by Amazon SES.
"""
send_args = {
    'Source': source,
    'Destination': destination.to_service_format(),
    'Template': template_name,
    'TemplateData': json.dumps(template_data)
}
if reply_tos is not None:
    send_args['ReplyToAddresses'] = reply_tos
try:
    response = self.ses_client.send templated_email(**send_args)
    message_id = response['MessageId']
    logger.info(
        "Sent templated mail %s from %s to %s.", message_id, source,
        destination.tos)
except ClientError:
    logger.exception(
        "Couldn't send templated mail from %s to %s.", source, destination.tos)
    raise
else:
    return message_id
```

Verify an email address with Amazon SES and send messages.

```
def usage_demo():
    print('*'*88)
    print("Welcome to the Amazon Simple Email Service (Amazon SES) email demo!")
    print('*'*88)

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    ses_client = boto3.client('ses')
    ses_identity = SesIdentity(ses_client)
    ses_mail_sender = SesMailSender(ses_client)
    ses_template = SesTemplate(ses_client)
    email = input(
        "Enter an email address to send mail with Amazon SES: ")
    status = ses_identity.get_identity_status(email)
    verified = status == 'Success'
    if not verified:
        answer = input(
            f"The address '{email}' is not verified with Amazon SES. Unless your "
            f"Amazon SES account is out of sandbox, you can send mail only from "
            f"and to verified accounts. Do you want to verify this account for use "
            f"with Amazon SES? If yes, the address will receive a verification "
            f"email (y/n): ")
        if answer.lower() == 'y':
            ses_identity.verify_email_identity(email)
            print(f"Follow the steps in the email to {email} to complete "
                  "verification.")
            print("Waiting for verification...")
            try:
                ses_identity.wait_until_identity_exists(email)
                print(f"Identity verified for {email}.")
                verified = True
            except WaiterError:
                print(f"Verification timeout exceeded. You must complete the "
                      f"steps in the email sent to {email} to verify the address.")

    if verified:
        test_message_text = "Hello from the Amazon SES mail demo!"
        test_message_html = "<p>Hello!</p><p>From the <b>Amazon SES</b> mail demo!</p>"
```

```
print(f"Sending mail from {email} to {email}.")  
ses_mail_sender.send_email(  
    email, SesDestination([email]), "Amazon SES demo",  
    test_message_text, test_message_html)  
input("Mail sent. Check your inbox and press Enter to continue.")  
  
template = {  
    'name': 'doc-example-template',  
    'subject': 'Example of an email template.',  
    'text': "This is what {{name}} will {{action}} if {{name}} can't display "  
        "HTML.",  
    'html': "<p><i>This</i> is what {{name}} will {{action}} if {{name}} " "  
        "<b>can</b> display HTML.</p>"}  
print("Creating a template and sending a templated email.")  
ses_template.create_template(**template)  
template_data = {'name': email.split('@')[0], 'action': 'read'}  
if ses_template.verify_tags(template_data):  
    ses_mail_sender.send_templated_email(  
        email, SesDestination([email]), ses_template.name(), template_data)  
    input("Mail sent. Check your inbox and press Enter to continue.")  
  
print("Sending mail through the Amazon SES SMTP server.")  
boto3_session = boto3.Session()  
region = boto3_session.region_name  
credentials = boto3_session.get_credentials()  
port = 587  
smtp_server = f'email-smtp.{region}.amazonaws.com'  
password = calculate_key(credentials.secret_key, region)  
message = """  
Subject: Hi there  
  
This message is sent from the Amazon SES SMTP mail demo."""  
context = ssl.create_default_context()  
with smtplib.SMTP(smtp_server, port) as server:  
    server.starttls(context=context)  
    server.login(credentials.access_key, password)  
    server.sendmail(email, email, message)  
print("Mail sent. Check your inbox!")  
  
if ses_template.template is not None:  
    print("Deleting demo template.")  
    ses_template.delete_template()  
if verified:  
    answer = input(f"Do you want to remove {email} from Amazon SES (y/n)? ")  
    if answer.lower() == 'y':  
        ses_identity.delete_identity(email)  
print("Thanks for watching!")  
print('*'*88)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateTemplate](#)
  - [DeleteIdentity](#)
  - [DeleteTemplate](#)
  - [GetIdentityVerificationAttributes](#)
  - [GetTemplate](#)
  - [ListIdentities](#)
  - [ListTemplates](#)
  - [SendEmail](#)
  - [SendTemplatedEmail](#)

- [UpdateTemplate](#)
- [VerifyDomainIdentity](#)
- [VerifyEmailIdentity](#)

## Amazon SNS examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Simple Notification Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4247\)](#)

## Actions

### Create a topic

The following code example shows how to create an Amazon SNS topic.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:  
    """Encapsulates Amazon SNS topic and subscription functions."""  
    def __init__(self, sns_resource):  
        """  
        :param sns_resource: A Boto3 Amazon SNS resource.  
        """  
        self.sns_resource = sns_resource  
  
    def create_topic(self, name):  
        """  
        Creates a notification topic.  
  
        :param name: The name of the topic to create.  
        :return: The newly created topic.  
        """  
        try:  
            topic = self.sns_resource.create_topic(Name=name)  
            logger.info("Created topic %s with ARN %s.", name, topic.arn)  
        except ClientError:  
            logger.exception("Couldn't create topic %s.", name)  
            raise  
        else:  
            return topic
```

- For API details, see [CreateTopic](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a subscription

The following code example shows how to delete an Amazon SNS subscription.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.", subscription.arn)
            raise
```

- For API details, see [Unsubscribe](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a topic

The following code example shows how to delete an Amazon SNS topic and all subscriptions to that topic.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
```

```
logger.exception("Couldn't delete topic %s.", topic.arn)
raise
```

- For API details, see [DeleteTopic](#) in *AWS SDK for Python (Boto3) API Reference*.

## List the subscribers of a topic

The following code example shows how to retrieve the list of subscribers of an Amazon SNS topic.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def list_subscriptions(self, topic=None):
        """
        Lists subscriptions for the current account, optionally limited to a
        specific topic.

        :param topic: When specified, only subscriptions to this topic are returned.
        :return: An iterator that yields the subscriptions.
        """
        try:
            if topic is None:
                subs_iter = self.sns_resource.subscriptions.all()
            else:
                subs_iter = topic.subscriptions.all()
                logger.info("Got subscriptions.")
        except ClientError:
            logger.exception("Couldn't get subscriptions.")
            raise
        else:
            return subs_iter
```

- For API details, see [ListSubscriptions](#) in *AWS SDK for Python (Boto3) API Reference*.

## List topics

The following code example shows how to list Amazon SNS topics.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:
```

```
"""Encapsulates Amazon SNS topic and subscription functions."""
def __init__(self, sns_resource):
    """
    :param sns_resource: A Boto3 Amazon SNS resource.
    """
    self.sns_resource = sns_resource

def list_topics(self):
    """
    Lists topics for the current account.

    :return: An iterator that yields the topics.
    """
    try:
        topics_iter = self.sns_resource.topics.all()
        logger.info("Got topics.")
    except ClientError:
        logger.exception("Couldn't get topics.")
        raise
    else:
        return topics_iter
```

- For API details, see [ListTopics](#) in *AWS SDK for Python (Boto3) API Reference*.

## Publish an SMS text message

The following code example shows how to publish SMS messages using Amazon SNS.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.

        :param phone_number: The phone number that receives the message. This must be
                            in E.164 format. For example, a United States phone
                            number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message)
            message_id = response['MessageId']
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
```

```
    else:  
        return message_id
```

- For API details, see [Publish](#) in [AWS SDK for Python \(Boto3\) API Reference](#).

## Publish to a topic

The following code example shows how to publish messages to an Amazon SNS topic.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Publish a message with attributes so that a subscription can filter based on attributes.

```
class SnsWrapper:  
    """Encapsulates Amazon SNS topic and subscription functions."""  
    def __init__(self, sns_resource):  
        """  
        :param sns_resource: A Boto3 Amazon SNS resource.  
        """  
        self.sns_resource = sns_resource  
  
    def publish_message(topic, message, attributes):  
        """  
        Publishes a message, with attributes, to a topic. Subscriptions can be filtered  
        based on message attributes so that a subscription receives messages only  
        when specified attributes are present.  
  
        :param topic: The topic to publish to.  
        :param message: The message to publish.  
        :param attributes: The key-value attributes to attach to the message. Values  
                          must be either `str` or `bytes`.  
        :return: The ID of the message.  
        """  
        try:  
            att_dict = {}  
            for key, value in attributes.items():  
                if isinstance(value, str):  
                    att_dict[key] = {'DataType': 'String', 'StringValue': value}  
                elif isinstance(value, bytes):  
                    att_dict[key] = {'DataType': 'Binary', 'BinaryValue': value}  
            response = topic.publish(Message=message, MessageAttributes=att_dict)  
            message_id = response['MessageId']  
            logger.info(  
                "Published message with attributes %s to topic %s.", attributes,  
                topic.arn)  
        except ClientError:  
            logger.exception("Couldn't publish message to topic %s.", topic.arn)  
            raise  
        else:  
            return message_id
```

Publish a message that takes different forms based on the protocol of the subscriber.

```
class SnsWrapper:  
    """Encapsulates Amazon SNS topic and subscription functions."""
```

```
def __init__(self, sns_resource):
    """
    :param sns_resource: A Boto3 Amazon SNS resource.
    """
    self.sns_resource = sns_resource

def publish_multi_message(
        topic, subject, default_message, sms_message, email_message):
    """
    Publishes a multi-format message to a topic. A multi-format message takes
    different forms based on the protocol of the subscriber. For example,
    an SMS subscriber might receive a short, text-only version of the message
    while an email subscriber could receive an HTML version of the message.

    :param topic: The topic to publish to.
    :param subject: The subject of the message.
    :param default_message: The default version of the message. This version is
                           sent to subscribers that have protocols that are not
                           otherwise specified in the structured message.
    :param sms_message: The version of the message sent to SMS subscribers.
    :param email_message: The version of the message sent to email subscribers.
    :return: The ID of the message.
    """
    try:
        message = {
            'default': default_message,
            'sms': sms_message,
            'email': email_message
        }
        response = topic.publish(
            Message=json.dumps(message), Subject=subject, MessageStructure='json')
        message_id = response['MessageId']
        logger.info("Published multi-format message to topic %s.", topic.arn)
    except ClientError:
        logger.exception("Couldn't publish message to topic %s.", topic.arn)
        raise
    else:
        return message_id
```

- For API details, see [Publish in AWS SDK for Python \(Boto3\) API Reference](#).

## Set a filter policy

The following code example shows how to set an Amazon SNS filter policy.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def add_subscription_filter(subscription, attributes):
        """
```

Adds a filter policy to a subscription. A filter policy is a key and a list of values that are allowed. When a message is published, it must have an attribute that passes the filter or it will not be sent to the subscription.

```
:param subscription: The subscription the filter policy is attached to.  
:param attributes: A dictionary of key-value pairs that define the filter.  
"""  
try:  
    att_policy = {key: [value] for key, value in attributes.items()}  
    subscription.set_attributes(  
        AttributeName='FilterPolicy', AttributeValue=json.dumps(att_policy))  
    logger.info("Added filter to subscription %s.", subscription.arn)  
except ClientError:  
    logger.exception(  
        "Couldn't add filter to subscription %s.", subscription.arn)  
    raise
```

- For API details, see [SetSubscriptionAttributes in AWS SDK for Python \(Boto3\) API Reference](#).

## Subscribe an email address to a topic

The following code example shows how to subscribe an email address to an Amazon SNS topic.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SnsWrapper:  
    """Encapsulates Amazon SNS topic and subscription functions."""  
    def __init__(self, sns_resource):  
        """  
        :param sns_resource: A Boto3 Amazon SNS resource.  
        """  
        self.sns_resource = sns_resource  
  
    def subscribe(topic, protocol, endpoint):  
        """  
        Subscribes an endpoint to the topic. Some endpoint types, such as email,  
        must be confirmed before their subscriptions are active. When a subscription  
        is not confirmed, its Amazon Resource Number (ARN) is set to  
        'PendingConfirmation'.  
  
        :param topic: The topic to subscribe to.  
        :param protocol: The protocol of the endpoint, such as 'sms' or 'email'.  
        :param endpoint: The endpoint that receives messages, such as a phone number  
                        (in E.164 format) for SMS messages, or an email address for  
                        email messages.  
        :return: The newly added subscription.  
        """  
        try:  
            subscription = topic.subscribe(  
                Protocol=protocol, Endpoint=endpoint, ReturnSubscriptionArn=True)  
            logger.info("Subscribed %s %s to topic %s.", protocol, endpoint, topic.arn)  
        except ClientError:  
            logger.exception(  
                "Couldn't subscribe %s %s to topic %s.", protocol, endpoint, topic.arn)  
            raise  
        else:  
            return subscription
```

- For API details, see [Subscribe](#) in *AWS SDK for Python (Boto3) API Reference*.

## Amazon SQS examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Simple Queue Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4254\)](#)
- [Scenarios \(p. 4259\)](#)

## Actions

### Create a queue

The following code example shows how to create an Amazon SQS queue.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_queue(name, attributes=None):
    """
    Creates an Amazon SQS queue.

    :param name: The name of the queue. This is part of the URL assigned to the queue.
    :param attributes: The attributes of the queue, such as maximum message size or
                      whether it's a FIFO queue.
    :return: A Queue object that contains metadata about the queue and that can be used
            to perform queue operations like sending and receiving messages.
    """
    if not attributes:
        attributes = {}

    try:
        queue = sqs.create_queue(
            QueueName=name,
            Attributes=attributes
        )
        logger.info("Created queue '%s' with URL=%s", name, queue.url)
    except ClientError as error:
        logger.exception("Couldn't create queue named '%s'.", name)
        raise error
    else:
        return queue
```

- For API details, see [CreateQueue](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a batch of messages from a queue

The following code example shows how to delete a batch of messages from an Amazon SQS queue.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_messages(queue, messages):
    """
    Delete a batch of messages from a queue in a single request.

    :param queue: The queue from which to delete the messages.
    :param messages: The list of messages to delete.
    :return: The response from SQS that contains the list of successful and failed
            message deletions.
    """
    try:
        entries = [
            {
                'Id': str(ind),
                'ReceiptHandle': msg.receipt_handle
            } for ind, msg in enumerate(messages)]
        response = queue.delete_messages(Entries=entries)
        if 'Successful' in response:
            for msg_meta in response['Successful']:
                logger.info("Deleted %s", messages[int(msg_meta['Id'])].receipt_handle)
        if 'Failed' in response:
            for msg_meta in response['Failed']:
                logger.warning(
                    "Could not delete %s",
                    messages[int(msg_meta['Id'])].receipt_handle
                )
    except ClientError:
        logger.exception("Couldn't delete messages from queue %s", queue)
    else:
        return response
```

- For API details, see [DeleteMessageBatch](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a message from a queue

The following code example shows how to delete a message from an Amazon SQS queue.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_message(message):
    """
    Delete a message from a queue. Clients must delete messages after they
    are received and processed to remove them from the queue.

    :param message: The message to delete. The message's queue URL is contained in
                    the message's metadata.
    :return: None
    """
```

```
"""
try:
    message.delete()
    logger.info("Deleted message: %s", message.message_id)
except ClientError as error:
    logger.exception("Couldn't delete message: %s", message.message_id)
    raise error
```

- For API details, see [DeleteMessage](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a queue

The following code example shows how to delete an Amazon SQS queue.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def remove_queue(queue):
    """
    Removes an SQS queue. When run against an AWS account, it can take up to
    60 seconds before the queue is actually deleted.

    :param queue: The queue to delete.
    :return: None
    """
    try:
        queue.delete()
        logger.info("Deleted queue with URL=%s.", queue.url)
    except ClientError as error:
        logger.exception("Couldn't delete queue with URL=%s!", queue.url)
        raise error
```

- For API details, see [DeleteQueue](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get the URL of a queue

The following code example shows how to get the URL of an Amazon SQS queue.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_queue(name):
    """
    Gets an SQS queue by name.

    :param name: The name that was used to create the queue.
    :return: A Queue object.
    """
    try:
        queue = sqs.get_queue_by_name(QueueName=name)
```

```
    logger.info("Got queue '%s' with URL=%s", name, queue.url)
except ClientError as error:
    logger.exception("Couldn't get queue named %s.", name)
    raise error
else:
    return queue
```

- For API details, see [GetQueueUrl](#) in *AWS SDK for Python (Boto3) API Reference*.

## List queues

The following code example shows how to list Amazon SQS queues.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_queues(prefix=None):
    """
    Gets a list of SQS queues. When a prefix is specified, only queues with names
    that start with the prefix are returned.

    :param prefix: The prefix used to restrict the list of returned queues.
    :return: A list of Queue objects.
    """
    if prefix:
        queue_iter = sqs.queues.filter(QueueNamePrefix=prefix)
    else:
        queue_iter = sqs.queues.all()
    queues = list(queue_iter)
    if queues:
        logger.info("Got queues: %s", ', '.join([q.url for q in queues]))
    else:
        logger.warning("No queues found.")
    return queues
```

- For API details, see [ListQueues](#) in *AWS SDK for Python (Boto3) API Reference*.

## Receive messages from a queue

The following code example shows how to receive messages from an Amazon SQS queue.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def receive_messages(queue, max_number, wait_time):
    """
    Receive a batch of messages in a single request from an SQS queue.

    :param queue: The queue from which to receive messages.
    :param max_number: The maximum number of messages to receive. The actual number
                       of messages received might be less.
    """
```

```
:param wait_time: The maximum time to wait (in seconds) before returning. When
    this number is greater than zero, long polling is used. This
    can result in reduced costs and fewer false empty responses.
:return: The list of Message objects received. These each contain the body
    of the message and metadata and custom attributes.
"""
try:
    messages = queue.receive_messages(
        MessageAttributeNames=['All'],
        MaxNumberOfMessages=max_number,
        WaitTimeSeconds=wait_time
    )
    for msg in messages:
        logger.info("Received message: %s: %s", msg.message_id, msg.body)
except ClientError as error:
    logger.exception("Couldn't receive messages from queue: %s", queue)
    raise error
else:
    return messages
```

- For API details, see [ReceiveMessage in AWS SDK for Python \(Boto3\) API Reference](#).

## Send a batch of messages to a queue

The following code example shows how to send a batch of messages to an Amazon SQS queue.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def send_messages(queue, messages):
    """
    Send a batch of messages in a single request to an SQS queue.
    This request may return overall success even when some messages were not sent.
    The caller must inspect the Successful and Failed lists in the response and
    resend any failed messages.

    :param queue: The queue to receive the messages.
    :param messages: The messages to send to the queue. These are simplified to
        contain only the message body and attributes.
    :return: The response from SQS that contains the list of successful and failed
        messages.
"""
try:
    entries = [
        {
            'Id': str(ind),
            'MessageBody': msg['body'],
            'MessageAttributes': msg['attributes']
        } for ind, msg in enumerate(messages)]
    response = queue.send_messages(Entries=entries)
    if 'Successful' in response:
        for msg_meta in response['Successful']:
            logger.info(
                "Message sent: %s: %s",
                msg_meta['MessageId'],
                messages[int(msg_meta['Id'])]['body']
            )
    if 'Failed' in response:
        for msg_meta in response['Failed']:
            logger.error(
                "Message failed: %s: %s",
                msg_meta['MessageId'],
                msg_meta['Reason']
            )

```

```
        logger.warning(
            "Failed to send: %s: %s",
            msg_meta['MessageId'],
            messages[int(msg_meta['Id'])]['body']
        )
    except ClientError as error:
        logger.exception("Send messages failed to queue: %s", queue)
        raise error
    else:
        return response
```

- For API details, see [SendMessageBatch](#) in *AWS SDK for Python (Boto3) API Reference*.

## Send a message to a queue

The following code example shows how to send a message to an Amazon SQS queue.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def send_message(queue, message_body, message_attributes=None):
    """
    Send a message to an Amazon SQS queue.

    :param queue: The queue that receives the message.
    :param message_body: The body text of the message.
    :param message_attributes: Custom attributes of the message. These are key-value
                              pairs that can be whatever you want.
    :return: The response from SQS that contains the assigned message ID.
    """
    if not message_attributes:
        message_attributes = {}

    try:
        response = queue.send_message(
            MessageBody=message_body,
            MessageAttributes=message_attributes
        )
    except ClientError as error:
        logger.exception("Send message failed: %s", message_body)
        raise error
    else:
        return response
```

- For API details, see [SendMessage](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Send and receive batches of messages

The following code example shows how to:

- Create an Amazon SQS queue.
- Send batches of messages to the queue.

- Receive batches of messages from the queue.
- Delete batches of messages from the queue.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create functions to wrap Amazon SQS message functions.

```
import logging
import sys

import boto3
from botocore.exceptions import ClientError

import queue_wrapper

logger = logging.getLogger(__name__)
sns = boto3.resource('sns')

def send_messages(queue, messages):
    """
    Send a batch of messages in a single request to an SQS queue.
    This request may return overall success even when some messages were not sent.
    The caller must inspect the Successful and Failed lists in the response and
    resend any failed messages.

    :param queue: The queue to receive the messages.
    :param messages: The messages to send to the queue. These are simplified to
                     contain only the message body and attributes.
    :return: The response from SQS that contains the list of successful and failed
            messages.
    """
    try:
        entries = []
        for ind, msg in enumerate(messages):
            entries.append({
                'Id': str(ind),
                'MessageBody': msg['body'],
                'MessageAttributes': msg['attributes']
            })
        response = queue.send_messages(Entries=entries)
        if 'Successful' in response:
            for msg_meta in response['Successful']:
                logger.info(
                    "Message sent: %s: %s",
                    msg_meta['MessageId'],
                    messages[int(msg_meta['Id'])]['body']
                )
        if 'Failed' in response:
            for msg_meta in response['Failed']:
                logger.warning(
                    "Failed to send: %s: %s",
                    msg_meta['MessageId'],
                    messages[int(msg_meta['Id'])]['body']
                )
    except ClientError as error:
        logger.exception("Send messages failed to queue: %s", queue)
        raise error
    else:
        return response

def receive_messages(queue, max_number, wait_time):
```

```

"""
Receive a batch of messages in a single request from an SQS queue.

:param queue: The queue from which to receive messages.
:param max_number: The maximum number of messages to receive. The actual number
                   of messages received might be less.
:param wait_time: The maximum time to wait (in seconds) before returning. When
                  this number is greater than zero, long polling is used. This
                  can result in reduced costs and fewer false empty responses.
:return: The list of Message objects received. These each contain the body
         of the message and metadata and custom attributes.
"""

try:
    messages = queue.receive_messages(
        MessageAttributeNames=['All'],
        MaxNumberOfMessages=max_number,
        WaitTimeSeconds=wait_time
    )
    for msg in messages:
        logger.info("Received message: %s: %s", msg.message_id, msg.body)
except ClientError as error:
    logger.exception("Couldn't receive messages from queue: %s", queue)
    raise error
else:
    return messages

def delete_messages(queue, messages):
    """
Delete a batch of messages from a queue in a single request.

:param queue: The queue from which to delete the messages.
:param messages: The list of messages to delete.
:return: The response from SQS that contains the list of successful and failed
        message deletions.
    """

    try:
        entries = [
            {
                'Id': str(ind),
                'ReceiptHandle': msg.receipt_handle
            } for ind, msg in enumerate(messages)]
        response = queue.delete_messages(Entries=entries)
        if 'Successful' in response:
            for msg_meta in response['Successful']:
                logger.info("Deleted %s", messages[int(msg_meta['Id'])].receipt_handle)
        if 'Failed' in response:
            for msg_meta in response['Failed']:
                logger.warning(
                    "Could not delete %s",
                    messages[int(msg_meta['Id'])].receipt_handle
                )
    except ClientError:
        logger.exception("Couldn't delete messages from queue %s", queue)
    else:
        return response

```

Use the wrapper functions to send and receive messages in batches.

```

def usage_demo():
    """
Shows how to:
* Read the lines from this Python file and send the lines in
  batches of 10 as messages to a queue.
* Receive the messages in batches until the queue is empty.
* Reassemble the lines of the file and verify they match the original file.

```

```
"""
def pack_message(msg_path, msg_body, msg_line):
    return {
        'body': msg_body,
        'attributes': {
            'path': {'StringValue': msg_path, 'DataType': 'String'},
            'line': {'StringValue': str(msg_line), 'DataType': 'String'}
        }
    }

def unpack_message(msg):
    return (msg.message_attributes['path']['StringValue'],
            msg.body,
            int(msg.message_attributes['line']['StringValue']))

print('*'*88)
print("Welcome to the Amazon Simple Queue Service (Amazon SQS) demo!")
print('*'*88)

queue = queue_wrapper.create_queue('sq-sqs-usage-demo-message-wrapper')

with open(__file__) as file:
    lines = file.readlines()

line = 0
batch_size = 10
received_lines = [None]*len(lines)
print(f"Sending file lines in batches of {batch_size} as messages.")
while line < len(lines):
    messages = [pack_message(__file__, lines[index], index)
                for index in range(line, min(line + batch_size, len(lines)))]
    line = line + batch_size
    send_messages(queue, messages)
    print('.')
    sys.stdout.flush()
print(f"Done. Sent {len(lines) - 1} messages.")

print(f"Receiving, handling, and deleting messages in batches of {batch_size}.")
more_messages = True
while more_messages:
    received_messages = receive_messages(queue, batch_size, 2)
    print('.')
    sys.stdout.flush()
    for message in received_messages:
        path, body, line = unpack_message(message)
        received_lines[line] = body
    if received_messages:
        delete_messages(queue, received_messages)
    else:
        more_messages = False
print('Done.')

if all([lines[index] == received_lines[index] for index in range(len(lines))]):
    print(f"Successfully reassembled all file lines!")
else:
    print(f"Uh oh, some lines were missed!")

queue.delete()

print("Thanks for watching!")
print('*'*88)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateQueue](#)

- [DeleteMessageBatch](#)
- [DeleteQueue](#)
- [ReceiveMessage](#)
- [SendMessageBatch](#)

## AWS STS examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with AWS Security Token Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4263\)](#)
- [Scenarios \(p. 4265\)](#)

## Actions

### Assume a role

The following code example shows how to assume a role with AWS STS.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Assume an IAM role that requires an MFA token and use temporary credentials to list Amazon S3 buckets for the account.

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
        grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual MFA
        device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp)
    temp_credentials = response['Credentials']
```

```
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        's3',
        aws_access_key_id=temp_credentials['AccessKeyId'],
        aws_secret_access_key=temp_credentials['SecretAccessKey'],
        aws_session_token=temp_credentials['SessionToken'])

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- For API details, see [AssumeRole](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a session token

The following code example shows how to get a session token with AWS STS and use it to perform a service action that requires an MFA token.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get a session token by passing an MFA token and use it to list Amazon S3 buckets for the account.

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp, sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual MFA
        device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp)
    else:
        response = sts_client.get_session_token()
    temp_credentials = response['Credentials']

    s3_resource = boto3.resource(
        's3',
        aws_access_key_id=temp_credentials['AccessKeyId'],
        aws_secret_access_key=temp_credentials['SecretAccessKey'],
        aws_session_token=temp_credentials['SessionToken'])

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- For API details, see [GetSessionToken](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Assume an IAM role that requires an MFA token

The following code example shows how to:

- Create an IAM role that grants permission to list Amazon S3 buckets.
- Create an IAM user that has permission to assume the role only when MFA credentials are provided.
- Register an MFA device for the user.
- Assume the role and use temporary credentials to list S3 buckets.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an IAM user, register an MFA device, and create a role that grants permission to list S3 buckets. The user has rights only to assume the role.

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual MFA device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role and requires MFA.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    For demonstration purposes, the user is created in the same account as the role,
    but in practice the user would likely be from another account.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
                         that has permissions to create users, roles, and policies
                         in the account.
    :return: The newly created user, user key, virtual MFA device, and role.
    """
    user = iam_resource.create_user(UserName=unique_name('user'))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name('mfa'))
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(f"Showing the QR code for the device. Scan this in the MFA app of your "
          f"choice.")
    with open('qr.png', 'wb') as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa()
```

```
    SerialNumber=virtual_mfa_device.serial_number,
    AuthenticationCode1=mfa_code_1,
    AuthenticationCode2=mfa_code_2)
os.remove(qr_file.name)
print(f"MFA device is registered with the user.")

user_key = user.create_access_key_pair()
print(f"Created access key pair for user.")

print(f"Wait for user to be ready.", end='')
progress_bar(10)

role = iam_resource.create_role(
    RoleName=unique_name('role'),
    AssumeRolePolicyDocument=json.dumps({
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Principal': {'AWS': user.arn},
                'Action': 'sts:AssumeRole',
                'Condition': {'Bool': {'aws:MultiFactorAuthPresent': True}}
            }
        ]
    })
)
print(f"Created role {role.name} that requires MFA.")

policy = iam_resource.create_policy(
    PolicyName=unique_name('policy'),
    PolicyDocument=json.dumps({
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Action': 's3>ListAllMyBuckets',
                'Resource': 'arn:aws:s3:::/*'
            }
        ]
    })
)
role.attach_policy(PolicyArn=policy.arn)
print(f"Created policy {policy.policy_name} and attached it to the role.")

user.create_policy(
    PolicyName=unique_name('user-policy'),
    PolicyDocument=json.dumps({
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Action': 'sts:AssumeRole',
                'Resource': role.arn
            }
        ]
    })
)
print(f"Created an inline policy for {user.name} that lets the user assume "
      f"the role.")

print("Give AWS time to propagate these new resources and connections.", end='')
progress_bar(10)

return user, user_key, virtual_mfa_device, role
```

Show that assuming the role without an MFA token is not allowed.

```
def try_to_assume_role_without_mfa(assume_role_arn, session_name, sts_client):
    """
    Shows that attempting to assume the role without sending MFA credentials results
    in an AccessDenied error.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role to assume.
    :param session_name: The name of the STS session.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    print(f"Trying to assume the role without sending MFA credentials...")
    try:
        sts_client.assume_role(RoleArn=assume_role_arn, RoleSessionName=session_name)
        raise RuntimeError("Expected AccessDenied error.")
    except ClientError as error:
        if error.response['Error']['Code'] == 'AccessDenied':
            print("Got AccessDenied.")
        else:
            raise
```

Assume the role that grants permission to list S3 buckets, passing the required MFA token, and show that buckets can be listed.

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                           grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual MFA
                             device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp)
    temp_credentials = response['Credentials']
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        's3',
        aws_access_key_id=temp_credentials['AccessKeyId'],
        aws_secret_access_key=temp_credentials['SecretAccessKey'],
        aws_session_token=temp_credentials['SessionToken'])

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

Destroy the resources created for the demo.

```
def teardown(user, virtual_mfa_device, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        policy_name = attached.policy_name
        role.detach_policy(PolicyArn=attached.arn)
        attached.delete()
        print(f"Detached and deleted {policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

Run this scenario by using the previously defined functions.

```
def usage_demo():
    """Drives the demonstration."""
    print('-'*88)
    print(f"Welcome to the AWS Security Token Service assume role demo, "
          f"starring multi-factor authentication (MFA)!")
    print('-'*88)
    iam_resource = boto3.resource('iam')
    user, user_key, virtual_mfa_device, role = setup(iam_resource)
    print(f"Created {user.name} and {role.name}.")
    try:
        sts_client = boto3.client(
            'sts', aws_access_key_id=user_key.id,
            aws_secret_access_key=user_key.secret)
        try_to_assume_role_without_mfa(role.arn, 'demo-sts-session', sts_client)
        mfa_totp = input('Enter the code from your registered MFA device: ')
        list_buckets_from_assumed_role_with_mfa(
            role.arn, 'demo-sts-session', virtual_mfa_device.serial_number,
            mfa_totp, sts_client)
    finally:
        teardown(user, virtual_mfa_device, role)
    print("Thanks for watching!")
```

- For API details, see [AssumeRole](#) in *AWS SDK for Python (Boto3) API Reference*.

## Construct a URL for federated users

The following code example shows how to:

- Create an IAM role that grants read-only access to the current account's Amazon S3 resources.
- Get a security token from the AWS federation endpoint.
- Construct a URL that can be used to access the console with federated credentials.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a role that grants read-only access to the current account's S3 resources.

```
def setup(iam_resource):
    """
    Creates a role that can be assumed by the current user.
    Attaches a policy that allows only Amazon S3 read-only access.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) instance
                         that has the permission to create a role.
    :return: The newly created role.
    """
    role = iam_resource.create_role(
        RoleName=unique_name('role'),
        AssumeRolePolicyDocument=json.dumps({
            'Version': '2012-10-17',
            'Statement': [
                {
                    'Effect': 'Allow',
                    'Principal': {'AWS': iam_resource.CurrentUser().arn},
                    'Action': 'sts:AssumeRole'
                }
            ]
        })
    role.attach_policy(PolicyArn='arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess')
    print(f"Created role {role.name}.")

    print("Give AWS time to propagate these new resources and connections.", end='')
    progress_bar(10)

    return role
```

Get a security token from the AWS federation endpoint and construct a URL that can be used to access the console with federated credentials.

```
def construct_federated_url(assume_role_arn, session_name, issuer, sts_client):
    """
    Constructs a URL that gives federated users direct access to the AWS Management
    Console.

    1. Acquires temporary credentials from AWS Security Token Service (AWS STS) that
       can be used to assume a role with limited permissions.
    2. Uses the temporary credentials to request a sign-in token from the
       AWS federation endpoint.
    3. Builds a URL that can be used in a browser to navigate to the AWS federation
       endpoint, includes the sign-in token for authentication, and redirects to
       the AWS Management Console with permissions defined by the role that was
       specified in step 1.

    :param assume_role_arn: The role that specifies the permissions that are granted.
                           The current user must have permission to assume the role.
    :param session_name: The name for the STS session.
    :param issuer: The organization that issues the URL.
    :param sts_client: A Boto3 STS instance that can assume the role.
    :return: The federated URL.
    """
    response = sts_client.assume_role(
```

```

RoleArn=assume_role_arn, RoleSessionName=session_name)
temp_credentials = response['Credentials']
print(f"Assumed role {assume_role_arn} and got temporary credentials.")

session_data = {
    'sessionId': temp_credentials['AccessKeyId'],
    'sessionKey': temp_credentials['SecretAccessKey'],
    'sessionToken': temp_credentials['SessionToken']
}
aws_federated_signin_endpoint = 'https://signin.aws.amazon.com/federation'

# Make a request to the AWS federation endpoint to get a sign-in token.
# The requests.get function URL-encodes the parameters and builds the query string
# before making the request.
response = requests.get(
    aws_federated_signin_endpoint,
    params={
        'Action': 'getSigninToken',
        'SessionDuration': str(datetime.timedelta(hours=12).seconds),
        'Session': json.dumps(session_data)
    })
signin_token = json.loads(response.text)
print(f"Got a sign-in token from the AWS sign-in federation endpoint.")

# Make a federated URL that can be used to sign into the AWS Management Console.
query_string = urllib.parse.urlencode({
    'Action': 'login',
    'Issuer': issuer,
    'Destination': 'https://console.aws.amazon.com/',
    'SigninToken': signin_token['SigninToken']
})
federated_url = f'{aws_federated_signin_endpoint}?{query_string}'
return federated_url

```

Destroy the resources created for the demo.

```

def teardown(role):
    """
    Removes all resources created during setup.

    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        role.detach_policy(PolicyArn=attached.arn)
        print(f"Detached {attached.policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")

```

Run this scenario by using the previously defined functions.

```

def usage_demo():
    """Drives the demonstration."""
    print('-'*88)
    print(f"Welcome to the AWS Security Token Service federated URL demo.")
    print('-'*88)
    iam_resource = boto3.resource('iam')
    role = setup(iam_resource)
    sts_client = boto3.client('sts')
    try:
        federated_url = construct_federated_url(
            role.arn, 'AssumeRoleDemoSession', 'example.org', sts_client)

```

```
print("Constructed a federated URL that can be used to connect to the "
      "AWS Management Console with role-defined permissions:")
print('-'*88)
print(federated_url)
print('-'*88)
_ = input("Copy and paste the above URL into a browser to open the AWS "
          "Management Console with limited permissions. When done, press "
          "'Enter' to clean up and complete this demo.")
finally:
    teardown(role)
    print("Thanks for watching!")
```

- For API details, see [AssumeRole](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a session token that requires an MFA token

The following code example shows how to:

- Create an IAM role that grants permission to list Amazon S3 buckets.
- Create an IAM user that has permission to assume the role only when MFA credentials are provided.
- Register an MFA device for the user.
- Provide MFA credentials to get a session token and use temporary credentials to list S3 buckets.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an IAM user, register an MFA device, and create a role that grants permission to let the user list S3 buckets only when MFA credentials are used.

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual multi-factor authentication (MFA) device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates an inline policy for the user that lets the user list Amazon S3 buckets,
    but only when MFA credentials are used.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
                         that has permissions to create users, MFA devices, and
                         policies in the account.
    :return: The newly created user, user key, and virtual MFA device.
    """
    user = iam_resource.create_user(UserName=unique_name('user'))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name('mfa'))
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(f"Showing the QR code for the device. Scan this in the MFA app of your "
```

```

f"choice.")
with open('qr.png', 'wb') as qr_file:
    qr_file.write(virtual_mfa_device.qr_code_png)
webbrowser.open(qr_file.name)

print(f"Enter two consecutive code from your MFA device.")
mfa_code_1 = input("Enter the first code: ")
mfa_code_2 = input("Enter the second code: ")
user.enable_mfa(
    SerialNumber=virtual_mfa_device.serial_number,
    AuthenticationCode1=mfa_code_1,
    AuthenticationCode2=mfa_code_2)
os.remove(qr_file.name)
print(f"MFA device is registered with the user.")

user_key = user.create_access_key_pair()
print(f"Created access key pair for user.")

print(f"Wait for user to be ready.", end='')
progress_bar(10)

user.create_policy(
    PolicyName=unique_name('user-policy'),
    PolicyDocument=json.dumps({
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Action': 's3>ListAllMyBuckets',
                'Resource': 'arn:aws:s3:::*',
                'Condition': {'Bool': {'aws:MultiFactorAuthPresent': True}}
            }
        ]
    })
)
print(f"Created an inline policy for {user.name} that lets the user list buckets, "
      f"but only when MFA credentials are present.")

print("Give AWS time to propagate these new resources and connections.", end='')
progress_bar(10)

return user, user_key, virtual_mfa_device

```

Get temporary session credentials by passing an MFA token, and use the credentials to list S3 buckets for the account.

```

def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp, sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual MFA
                             device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp)
    else:
        response = sts_client.get_session_token()

```

```
temp_credentials = response['Credentials']

s3_resource = boto3.resource(
    's3',
    aws_access_key_id=temp_credentials['AccessKeyId'],
    aws_secret_access_key=temp_credentials['SecretAccessKey'],
    aws_session_token=temp_credentials['SessionToken'])

print(f"Buckets for the account:")
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

Destroy the resources created for the demo.

```
def teardown(user, virtual_mfa_device):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo MFA device.
    """
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

Run this scenario by using the previously defined functions.

```
def usage_demo():
    """Drives the demonstration."""
    print('-'*88)
    print(f"Welcome to the AWS Security Token Service assume role demo, "
          f"starring multi-factor authentication (MFA)!")
    print('-'*88)
    iam_resource = boto3.resource('iam')
    user, user_key, virtual_mfa_device = setup(iam_resource)
    try:
        sts_client = boto3.client(
            'sts', aws_access_key_id=user_key.id,
            aws_secret_access_key=user_key.secret)
        try:
            print("Listing buckets without specifying MFA credentials.")
            list_buckets_with_session_token_with_mfa(None, None, sts_client)
        except ClientError as error:
            if error.response['Error']['Code'] == 'AccessDenied':
                print("Got expected AccessDenied error.")
            mfa_totp = input('Enter the code from your registered MFA device: ')
            list_buckets_with_session_token_with_mfa(
                virtual_mfa_device.serial_number, mfa_totp, sts_client)
    finally:
        teardown(user, virtual_mfa_device)
        print("Thanks for watching!")
```

- For API details, see [GetSessionToken](#) in *AWS SDK for Python (Boto3) API Reference*.

## Secrets Manager examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with AWS Secrets Manager.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4274\)](#)
- [Scenarios \(p. 4280\)](#)

## Actions

### Create a secret

The following code example shows how to create a Secrets Manager secret.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:  
    """Encapsulates Secrets Manager functions."""  
    def __init__(self, secretsmanager_client):  
        """  
        :param secretsmanager_client: A Boto3 Secrets Manager client.  
        """  
        self.secretsmanager_client = secretsmanager_client  
        self.name = None  
  
    def create(self, name, secret_value):  
        """  
        Creates a new secret. The secret value can be a string or bytes.  
  
        :param name: The name of the secret to create.  
        :param secret_value: The value of the secret.  
        :return: Metadata about the newly created secret.  
        """  
        self._clear()  
        try:  
            kwargs = {'Name': name}  
            if isinstance(secret_value, str):  
                kwargs['SecretString'] = secret_value  
            elif isinstance(secret_value, bytes):  
                kwargs['SecretBinary'] = secret_value  
            response = self.secretsmanager_client.create_secret(**kwargs)  
            self.name = name  
            logger.info("Created secret %s.", name)  
        except ClientError:  
            logger.exception("Couldn't get secret %s.", name)  
            raise  
        else:
```

```
    return response
```

- For API details, see [CreateSecret](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a secret

The following code example shows how to delete a Secrets Manager secret.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:  
    """Encapsulates Secrets Manager functions."""  
    def __init__(self, secretsmanager_client):  
        """  
        :param secretsmanager_client: A Boto3 Secrets Manager client.  
        """  
        self.secretsmanager_client = secretsmanager_client  
        self.name = None  
  
    def delete(self, without_recovery):  
        """  
        Deletes the secret.  
  
        :param without_recovery: Permanently deletes the secret immediately when True;  
                               otherwise, the deleted secret can be restored within  
                               the recovery window. The default recovery window is  
                               30 days.  
        """  
        if self.name is None:  
            raise ValueError  
  
        try:  
            self.secretsmanager_client.delete_secret(  
                SecretId=self.name, ForceDeleteWithoutRecovery=without_recovery)  
            logger.info("Deleted secret %s.", self.name)  
            self._clear()  
        except ClientError:  
            logger.exception("Deleted secret %s.", self.name)  
            raise
```

- For API details, see [DeleteSecret](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a secret

The following code example shows how to describe a Secrets Manager secret.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:
```

```
"""Encapsulates Secrets Manager functions."""
def __init__(self, secretsmanager_client):
    """
    :param secretsmanager_client: A Boto3 Secrets Manager client.
    """
    self.secretsmanager_client = secretsmanager_client
    self.name = None

    def describe(self, name=None):
        """
        Gets metadata about a secret.

        :param name: The name of the secret to load. If `name` is None, metadata about
                     the current secret is retrieved.
        :return: Metadata about the secret.
        """
        if self.name is None and name is None:
            raise ValueError
        if name is None:
            name = self.name
        self._clear()
        try:
            response = self.secretsmanager_client.describe_secret(SecretId=name)
            self.name = name
            logger.info("Got secret metadata for %s.", name)
        except ClientError:
            logger.exception("Couldn't get secret metadata for %s.", name)
            raise
        else:
            return response
```

- For API details, see [DescribeSecret](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a random password

The following code example shows how to get a random password from Secrets Manager.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:
    """Encapsulates Secrets Manager functions."""
    def __init__(self, secretsmanager_client):
        """
        :param secretsmanager_client: A Boto3 Secrets Manager client.
        """
        self.secretsmanager_client = secretsmanager_client
        self.name = None

        def get_random_password(self, pw_length):
            """
            Gets a randomly generated password.

            :param pw_length: The length of the password.
            :return: The generated password.
            """
            try:
                response = self.secretsmanager_client.get_random_password(
                    PasswordLength=pw_length)
```

```
        password = response['RandomPassword']
        logger.info("Got random password.")
    except ClientError:
        logger.exception("Couldn't get random password.")
        raise
    else:
        return password
```

- For API details, see [GetRandomPassword](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a secret value

The following code example shows how to get a Secrets Manager secret value.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:
    """Encapsulates Secrets Manager functions."""
    def __init__(self, secretsmanager_client):
        """
        :param secretsmanager_client: A Boto3 Secrets Manager client.
        """
        self.secretsmanager_client = secretsmanager_client
        self.name = None

    def get_value(self, stage=None):
        """
        Gets the value of a secret.

        :param stage: The stage of the secret to retrieve. If this is None, the
                      current stage is retrieved.
        :return: The value of the secret. When the secret is a string, the value is
                 contained in the `SecretString` field. When the secret is bytes,
                 it is contained in the `SecretBinary` field.
        """
        if self.name is None:
            raise ValueError

        try:
            kwargs = {'SecretId': self.name}
            if stage is not None:
                kwargs['VersionStage'] = stage
            response = self.secretsmanager_client.get_secret_value(**kwargs)
            logger.info("Got value for secret %s.", self.name)
        except ClientError:
            logger.exception("Couldn't get value for secret %s.", self.name)
            raise
        else:
            return response
```

- For API details, see [GetSecretValue](#) in *AWS SDK for Python (Boto3) API Reference*.

## List secrets

The following code example shows how to list Secrets Manager secrets.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:  
    """Encapsulates Secrets Manager functions."""  
    def __init__(self, secretsmanager_client):  
        """  
        :param secretsmanager_client: A Boto3 Secrets Manager client.  
        """  
        self.secretsmanager_client = secretsmanager_client  
        self.name = None  
  
    def list(self, max_results):  
        """  
        Lists secrets for the current account.  
  
        :param max_results: The maximum number of results to return.  
        :return: Yields secrets one at a time.  
        """  
        try:  
            paginator = self.secretsmanager_client.getPaginator('list_secrets')  
            for page in paginator.paginate(  
                PaginationConfig={'MaxItems': max_results}):  
                for secret in page['SecretList']:  
                    yield secret  
        except ClientError:  
            logger.exception("Couldn't list secrets.")  
            raise
```

- For API details, see [ListSecrets](#) in *AWS SDK for Python (Boto3) API Reference*.

## Put a value in a secret

The following code example shows how to put a value in a Secrets Manager secret.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:  
    """Encapsulates Secrets Manager functions."""  
    def __init__(self, secretsmanager_client):  
        """  
        :param secretsmanager_client: A Boto3 Secrets Manager client.  
        """  
        self.secretsmanager_client = secretsmanager_client  
        self.name = None  
  
    def put_value(self, secret_value, stages=None):  
        """  
        Puts a value into an existing secret. When no stages are specified, the  
        value is set as the current ('AWSCURRENT') stage and the previous value is  
        moved to the 'AWSPREVIOUS' stage. When a stage is specified that already  
        exists, the stage is associated with the new value and removed from the old  
        value.  
        """
```

```
:param secret_value: The value to add to the secret.
:param stages: The stages to associate with the secret.
:return: Metadata about the secret.
"""
if self.name is None:
    raise ValueError

try:
    kwargs = {'SecretId': self.name}
    if isinstance(secret_value, str):
        kwargs['SecretString'] = secret_value
    elif isinstance(secret_value, bytes):
        kwargs['SecretBinary'] = secret_value
    if stages is not None:
        kwargs['VersionStages'] = stages
    response = self.secretsmanager_client.put_secret_value(**kwargs)
    logger.info("Value put in secret %s.", self.name)
except ClientError:
    logger.exception("Couldn't put value in secret %s.", self.name)
    raise
else:
    return response
```

- For API details, see [PutSecretValue](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update the stage of a secret version

The following code example shows how to update the stage of a Secrets Manager secret version.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SecretsManagerSecret:
    """Encapsulates Secrets Manager functions."""
    def __init__(self, secretsmanager_client):
        """
        :param secretsmanager_client: A Boto3 Secrets Manager client.
        """
        self.secretsmanager_client = secretsmanager_client
        self.name = None

    def update_version_stage(self, stage, remove_from, move_to):
        """
        Updates the stage associated with a version of the secret.

        :param stage: The stage to update.
        :param remove_from: The ID of the version to remove the stage from.
        :param move_to: The ID of the version to add the stage to.
        :return: Metadata about the secret.
        """
        if self.name is None:
            raise ValueError

        try:
            response = self.secretsmanager_client.update_secret_version_stage(
                SecretId=self.name, VersionStage=stage,
                RemoveFromVersionId=remove_from,
                MoveToVersionId=move_to)
```

```
        logger.info("Updated version stage %s for secret %s.", stage, self.name)
    except ClientError:
        logger.exception(
            "Couldn't update version stage %s for secret %s.", stage, self.name)
        raise
    else:
        return response
```

- For API details, see [UpdateSecretVersionStage in AWS SDK for Python \(Boto3\) API Reference](#).

## Scenarios

### Create and manage a secret

The following code example shows how to:

- Create a Secrets Manager secret.
- Generate a random password and update a secret.
- Get current and previous values of a secret.
- Store binary data in a secret.
- Delete a secret.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a secret, update its value and stage, and delete the secret.

```
def create_and_manage_secret_demo():
    """
    Shows how to use AWS Secrets Manager to create a secret, update its value and
    stage, and delete it.
    """
    secret = SecretsManagerSecret(boto3.client('secretsmanager'))

    print("Create a secret.")
    secret.create("doc-example-secretsmanager-secret", "Shh, don't tell.")
    print("Get secret value.")
    value = secret.get_value()
    print(f"Secret value: {value['SecretString']}")

    print("Get a random password.")
    password = secret.get_random_password(20)
    print(f"Got password: {password}")

    print("Put password as new secret value.")
    secret.put_value(password)

    print("Get current and previous values.")
    current = secret.get_value()
    previous = secret.get_value('AWSPREVIOUS')
    print(f"Current: {current['SecretString']}")
    print(f"Previous: {previous['SecretString']}")

    byteval = base64.b64encode("I'm a Base64 string!".encode('utf-8'))
    stage = 'CUSTOM_STAGE'
    print(f"Put byte value with a custom stage '{stage}'")
    secret.put_value(byteval, [stage])
    time.sleep(1)

    print(f"Get secret value associated with stage '{stage}'")
    got_val = secret.get_value(stage)
```

```
    print(f"Raw bytes value: {got_val['SecretBinary']}")  
    print(f"Decoded value:  
{base64.b64decode(got_val['SecretBinary']).decode('utf-8')})")  
    pprint(secret.describe())  
    print("List 10 secrets for the account.")  
    for sec in secret.list(10):  
        print(f"Name: {sec['Name']}")  
    print("Delete the secret.")  
    secret.delete(True)
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateSecret](#)
  - [DeleteSecret](#)
  - [DescribeSecret](#)
  - [GetRandomPassword](#)
  - [GetSecretValue](#)
  - [ListSecrets](#)
  - [PutSecretValue](#)

## Step Functions examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with AWS Step Functions.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4281\)](#)

## Actions

### Create a state machine

The following code example shows how to create a Step Functions state machine.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class StepFunctionsStateMachine:  
    """Encapsulates Step Functions state machine functions."""  
    def __init__(self, stepfunctions_client):  
        """  
        :param stepfunctions_client: A Boto3 Step Functions client.  
        """  
        self.stepfunctions_client = stepfunctions_client  
        self.state_machine_name = None  
        self.state_machine_arn = None  
  
    def create(self, name, definition, role_arn):  
        """
```

```
Creates a new state machine.

:param name: The name of the new state machine.
:param definition: A dict that contains all of the state and flow control
                   information. The dict is translated to JSON before it is
                   uploaded.
:param role_arn: A role that grants Step Functions permission to access any
                  AWS services that are specified in the definition.
:return: The Amazon Resource Name (ARN) of the new state machine.
"""
try:
    response = self.stepfunctions_client.create_state_machine(
        name=name, definition=json.dumps(definition), roleArn=role_arn)
    self.state_machine_name = name
    self.state_machine_arn = response['stateMachineArn']
    logger.info(
        "Created state machine %s. ARN is %s.", name, self.state_machine_arn)
except ClientError:
    logger.exception("Couldn't create state machine %s.", name)
    raise
else:
    return self.state_machine_arn
```

- For API details, see [CreateStateMachine](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a state machine

The following code example shows how to delete a Step Functions state machine.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class StepFunctionsStateMachine:
    """Encapsulates Step Functions state machine functions."""
    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client
        self.state_machine_name = None
        self.state_machine_arn = None

    def delete(self):
        """
        Deletes a state machine and all associated run information.
        """
        if self.state_machine_arn is None:
            raise ValueError
        try:
            self.stepfunctions_client.delete_state_machine(
                stateMachineArn=self.state_machine_arn)
            logger.info("Deleted state machine %s.", self.state_machine_name)
            self._clear()
        except ClientError:
            logger.exception(
                "Couldn't delete state machine %s.", self.state_machine_name)
            raise
```

- For API details, see [DeleteStateMachine](#) in *AWS SDK for Python (Boto3) API Reference*.

## Describe a state machine

The following code example shows how to describe a Step Functions state machine.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class StepFunctionsStateMachine:  
    """Encapsulates Step Functions state machine functions."""  
    def __init__(self, stepfunctions_client):  
        """  
        :param stepfunctions_client: A Boto3 Step Functions client.  
        """  
        self.stepfunctions_client = stepfunctions_client  
        self.state_machine_name = None  
        self.state_machine_arn = None  
  
    def describe(self):  
        """  
        Gets metadata about a state machine.  
  
        :return: The metadata about the state machine.  
        """  
        if self.state_machine_arn is None:  
            raise ValueError  
        try:  
            response = self.stepfunctions_client.describe_state_machine(  
                stateMachineArn=self.state_machine_arn)  
            logger.info("Got metadata for state machine %s.", self.state_machine_name)  
        except ClientError:  
            logger.exception(  
                "Couldn't get metadata for state machine %s.", self.state_machine_name)  
        else:  
            return response
```

- For API details, see [DescribeStateMachine](#) in *AWS SDK for Python (Boto3) API Reference*.

## List state machine runs

The following code example shows how to list Step Functions state machine runs.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class StepFunctionsStateMachine:  
    """Encapsulates Step Functions state machine functions."""  
    def __init__(self, stepfunctions_client):  
        """  
        :param stepfunctions_client: A Boto3 Step Functions client.  
        """
```

```
self.stepfunctions_client = stepfunctions_client
self.state_machine_name = None
self.state_machine_arn = None

def list_runs(self, run_status=None):
    """
    Lists the runs for the state machine.

    :param run_status: When specified, only lists runs that have the specified
                       status. Otherwise, all runs are listed.
    :return: The list of runs.
    """
    if self.state_machine_arn is None:
        raise ValueError
    try:
        kwargs = {'stateMachineArn': self.state_machine_arn}
        if run_status is not None:
            kwargs['statusFilter'] = run_status
        response = self.stepfunctions_client.list_executions(**kwargs)
        runs = response['executions']
        logger.info(
            "Got %s runs for state machine %s.", len(runs),
            self.state_machine_name)
    except ClientError:
        logger.exception(
            "Couldn't get runs for state machine %s.", self.state_machine_name)
        raise
    else:
        return runs
```

- For API details, see [ListExecutions](#) in *AWS SDK for Python (Boto3) API Reference*.

## List state machines

The following code example shows how to list Step Functions state machines.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Find a state machine by name by searching the list of state machines for the account.

```
class StepFunctionsStateMachine:
    """
    Encapsulates Step Functions state machine functions.
    """
    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client
        self.state_machine_name = None
        self.state_machine_arn = None

    def find(self, state_machine_name):
        """
        Finds a state machine by name. This function iterates the state machines for
        the current account until it finds a match and returns the first matching
        state machine.

        :param state_machine_name: The name of the state machine to find.
        :return: The ARN of the named state machine when found; otherwise, None.
        """
```

```
"""
self._clear()
try:
    paginator = self.stepfunctions_client.getPaginator('list_state_machines')
    for page in paginator.paginate():
        for machine in page['stateMachines']:
            if machine['name'] == state_machine_name:
                self.state_machine_name = state_machine_name
                self.state_machine_arn = machine['stateMachineArn']
                break
            if self.state_machine_arn is not None:
                break
    if self.state_machine_arn is not None:
        logger.info(
            "Found state machine %s with ARN %s.", self.state_machine_name,
            self.state_machine_arn)
    else:
        logger.info("Couldn't find state machine %s.", state_machine_name)
except ClientError:
    logger.exception("Couldn't find state machine %s.", state_machine_name)
    raise
else:
    return self.state_machine_arn
```

- For API details, see [ListStateMachines](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start a state machine run

The following code example shows how to start a Step Functions state machine run.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class StepFunctionsStateMachine:
    """Encapsulates Step Functions state machine functions."""
    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client
        self.state_machine_name = None
        self.state_machine_arn = None

    def start_run(self, run_name, run_input=None):
        """
        Starts a run with the current state definition.

        :param run_name: The name of the run. This name must be unique for all runs
                         for the state machine.
        :param run_input: Data that is passed as input to the run.
        :return: The ARN of the run.
        """
        if self.state_machine_arn is None:
            raise ValueError
        try:
            kwargs = {'stateMachineArn': self.state_machine_arn, 'name': run_name}
            if run_input is not None:
                kwargs['input'] = json.dumps(run_input)
            response = self.stepfunctions_client.start_execution(**kwargs)
        except ClientError:
            logger.exception("Couldn't start state machine run %s.", run_name)
            raise
        return response['executionArn']
```

```
        run_arn = response['executionArn']
        logger.info("Started run %s. ARN is %s.", run_name, run_arn)
    except ClientError:
        logger.exception("Couldn't start run %s.", run_name)
        raise
    else:
        return run_arn
```

- For API details, see [StartExecution](#) in *AWS SDK for Python (Boto3) API Reference*.

## Stop a state machine run

The following code example shows how to stop a Step Functions state machine run.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class StepFunctionsStateMachine:
    """Encapsulates Step Functions state machine functions."""
    def __init__(self, stepfunctions_client):
        """
        :param stepfunctions_client: A Boto3 Step Functions client.
        """
        self.stepfunctions_client = stepfunctions_client
        self.state_machine_name = None
        self.state_machine_arn = None

    def stop_run(self, run_arn, cause):
        """
        Stops a run.

        :param run_arn: The run to stop.
        :param cause: A description of why the run was stopped.
        """
        try:
            self.stepfunctions_client.stop_execution(executionArn=run_arn, cause=cause)
            logger.info("Stopping run %s.", run_arn)
        except ClientError:
            logger.exception("Couldn't stop run %s.", run_arn)
            raise
```

- For API details, see [StopExecution](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update a state machine

The following code example shows how to update a Step Functions state machine.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class StepFunctionsStateMachine:
    """Encapsulates Step Functions state machine functions."""
```

```
def __init__(self, stepfunctions_client):
    """
    :param stepfunctions_client: A Boto3 Step Functions client.
    """
    self.stepfunctions_client = stepfunctions_client
    self.state_machine_name = None
    self.state_machine_arn = None

def update(self, definition, role_arn=None):
    """
    Updates an existing state machine. Any runs currently operating do not update
    until they are stopped.

    :param definition: A dict that contains all of the state and flow control
                       information for the state machine. This completely replaces
                       the existing definition.
    :param role_arn: A role that grants Step Functions permission to access any
                     AWS services that are specified in the definition.
    """
    if self.state_machine_arn is None:
        raise ValueError
    try:
        kwargs = {
            'stateMachineArn': self.state_machine_arn,
            'definition': json.dumps(definition)}
        if role_arn is not None:
            kwargs['roleArn'] = role_arn
        self.stepfunctions_client.update_state_machine(**kwargs)
        logger.info("Updated state machine %s.", self.state_machine_name)
    except ClientError:
        logger.exception(
            "Couldn't update state machine %s.", self.state_machine_name)
        raise
```

- For API details, see [UpdateStateMachine](#) in *AWS SDK for Python (Boto3) API Reference*.

## Amazon Textract examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Textract.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4287\)](#)

## Actions

### Analyze a document

The following code example shows how to analyze a document using Amazon Textract.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def analyze_file(
        self, feature_types, *, document_file_name=None, document_bytes=None):
        """
        Detects text and additional elements, such as forms or tables, in a local image
        file or from in-memory byte data.
        The image must be in PNG or JPG format.

        :param feature_types: The types of additional document features to detect.
        :param document_file_name: The name of a document image file.
        :param document_bytes: In-memory byte data of a document image.
        :return: The response from Amazon Textract, including a list of blocks
            that describe elements detected in the image.
        """
        if document_file_name is not None:
            with open(document_file_name, 'rb') as document_file:
                document_bytes = document_file.read()
        try:
            response = self.textract_client.analyze_document(
                Document={'Bytes': document_bytes}, FeatureTypes=feature_types)
            logger.info(
                "Detected %s blocks.", len(response['Blocks']))
        except ClientError:
            logger.exception("Couldn't detect text.")
            raise
        else:
            return response
```

- For API details, see [AnalyzeDocument](#) in *AWS SDK for Python (Boto3) API Reference*.

## Detect text in a document

The following code example shows how to detect text in a document using Amazon Textract.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
```

```
self.s3_resource = s3_resource
self.sqs_resource = sqs_resource

def detect_file_text(self, *, document_file_name=None, document_bytes=None):
    """
    Detects text elements in a local image file or from in-memory byte data.
    The image must be in PNG or JPG format.

    :param document_file_name: The name of a document image file.
    :param document_bytes: In-memory byte data of a document image.
    :return: The response from Amazon Textract, including a list of blocks
            that describe elements detected in the image.
    """
    if document_file_name is not None:
        with open(document_file_name, 'rb') as document_file:
            document_bytes = document_file.read()
    try:
        response = self.textract_client.detect_document_text(
            Document={'Bytes': document_bytes})
        logger.info(
            "Detected %s blocks.", len(response['Blocks']))
    except ClientError:
        logger.exception("Couldn't detect text.")
        raise
    else:
        return response
```

- For API details, see [DetectDocumentText](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get data about a document analysis job

The following code example shows how to get data about an Amazon Textract document analysis job.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def get_analysis_job(self, job_id):
        """
        Gets data for a previously started detection job that includes additional
        elements.

        :param job_id: The ID of the job to retrieve.
        :return: The job data, including a list of blocks that describe elements
                detected in the image.
        """
        try:
```

```
        response = self.textract_client.get_document_analysis(
            JobId=job_id)
        job_status = response['JobStatus']
        logger.info("Job %s status is %s.", job_id, job_status)
    except ClientError:
        logger.exception("Couldn't get data for job %s.", job_id)
        raise
    else:
        return response
```

- For API details, see [GetDocumentAnalysis](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start asynchronous analysis of a document

The following code example shows how to start asynchronous analysis of a document using Amazon Textract.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Start an asynchronous job to analyze a document.

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def start_analysis_job(
        self, bucket_name, document_file_name, feature_types, sns_topic_arn,
        sns_role_arn):
        """
        Starts an asynchronous job to detect text and additional elements, such as
        forms or tables, in an image stored in an Amazon S3 bucket. Textract publishes
        a notification to the specified Amazon SNS topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the image.
        :param document_file_name: The name of the document image stored in Amazon S3.
        :param feature_types: The types of additional document features to detect.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS topic
            where job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management (IAM)
            role that can be assumed by Textract and grants permission
            to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_analysis(
                DocumentLocation={
                    'S3Object': {'Bucket': bucket_name, 'Name': document_file_name}},
                NotificationChannel={
                    'SNSTopicArn': sns_topic_arn, 'RoleArn': sns_role_arn},
```

```
        FeatureTypes=feature_types)
    job_id = response['JobId']
    logger.info(
        "Started text analysis job %s on %s.", job_id, document_file_name)
except ClientError:
    logger.exception("Couldn't analyze text in %s.", document_file_name)
    raise
else:
    return job_id
```

- For API details, see [StartDocumentAnalysis](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start asynchronous text detection

The following code example shows how to start asynchronous text detection in a document using Amazon Textract.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Start an asynchronous job to detect text in a document.

```
class TextractWrapper:
    """Encapsulates Textract functions."""
    def __init__(self, textract_client, s3_resource, sqs_resource):
        """
        :param textract_client: A Boto3 Textract client.
        :param s3_resource: A Boto3 Amazon S3 resource.
        :param sqs_resource: A Boto3 Amazon SQS resource.
        """
        self.textract_client = textract_client
        self.s3_resource = s3_resource
        self.sqs_resource = sqs_resource

    def start_detection_job(
            self, bucket_name, document_file_name, sns_topic_arn, sns_role_arn):
        """
        Starts an asynchronous job to detect text elements in an image stored in an
        Amazon S3 bucket. Textract publishes a notification to the specified Amazon SNS
        topic when the job completes.
        The image must be in PNG, JPG, or PDF format.

        :param bucket_name: The name of the Amazon S3 bucket that contains the image.
        :param document_file_name: The name of the document image stored in Amazon S3.
        :param sns_topic_arn: The Amazon Resource Name (ARN) of an Amazon SNS topic
            where the job completion notification is published.
        :param sns_role_arn: The ARN of an AWS Identity and Access Management (IAM)
            role that can be assumed by Textract and grants permission
            to publish to the Amazon SNS topic.
        :return: The ID of the job.
        """
        try:
            response = self.textract_client.start_document_text_detection(
                DocumentLocation={
                    'S3Object': {'Bucket': bucket_name, 'Name': document_file_name}},
                NotificationChannel={
                    'SNSTopicArn': sns_topic_arn, 'RoleArn': sns_role_arn})
            job_id = response['JobId']
            logger.info(
```

```
        "Started text detection job %s on %s.", job_id, document_file_name)
except ClientError:
    logger.exception("Couldn't detect text in %s.", document_file_name)
    raise
else:
    return job_id
```

- For API details, see [StartDocumentTextDetection](#) in *AWS SDK for Python (Boto3) API Reference*.

## Amazon Transcribe examples using SDK for Python (Boto3)

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Python (Boto3) with Amazon Transcribe.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4292\)](#)
- [Scenarios \(p. 4298\)](#)

## Actions

### Create a custom vocabulary

The following code example shows how to create a custom Amazon Transcribe vocabulary.

#### SDK for Python (Boto3)

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def create_vocabulary(
    vocabulary_name, language_code, transcribe_client,
    phrases=None, table_uri=None):
    """
    Creates a custom vocabulary that can be used to improve the accuracy of
    transcription jobs. This function returns as soon as the vocabulary processing
    is started. Call get_vocabulary to get the current status of the vocabulary.
    The vocabulary is ready to use when its status is 'READY'.

    :param vocabulary_name: The name of the custom vocabulary.
    :param language_code: The language code of the vocabulary.
        For example, en-US or nl-NL.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in the
        vocabulary.
    :return: Information about the newly created vocabulary.
    """
    try:
        vocab_args = {'VocabularyName': vocabulary_name, 'LanguageCode': language_code}
        if phrases is not None:
```

```
    vocab_args['Phrases'] = phrases
    elif table_uri is not None:
        vocab_args['VocabularyFileUri'] = table_uri
    response = transcribe_client.create_vocabulary(**vocab_args)
    logger.info("Created custom vocabulary %s.", response['VocabularyName'])
except ClientError:
    logger.exception("Couldn't create custom vocabulary %s.", vocabulary_name)
    raise
else:
    return response
```

- For API details, see [CreateVocabulary](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a custom vocabulary

The following code example shows how to delete a custom Amazon Transcribe vocabulary.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise
```

- For API details, see [DeleteVocabulary](#) in *AWS SDK for Python (Boto3) API Reference*.

## Delete a transcription job

The following code example shows how to delete an Amazon Transcribe transcription job.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def delete_job(job_name, transcribe_client):
    """
    Deletes a transcription job. This also deletes the transcript associated with
    the job.

    :param job_name: The name of the job to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
```

```
"""
try:
    transcribe_client.delete_transcription_job(
        TranscriptionJobName=job_name)
    logger.info("Deleted job %s.", job_name)
except ClientError:
    logger.exception("Couldn't delete job %s.", job_name)
    raise
```

- For API details, see [DeleteTranscriptionJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a custom vocabulary

The following code example shows how to get a custom Amazon Transcribe vocabulary.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_vocabulary(vocabulary_name, transcribe_client):
    """
    Gets information about a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: Information about the vocabulary.
    """
    try:
        response = transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Got vocabulary %s.", response['VocabularyName'])
    except ClientError:
        logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
        raise
    else:
        return response
```

- For API details, see [GetVocabulary](#) in *AWS SDK for Python (Boto3) API Reference*.

## Get a transcription job

The following code example shows how to get an Amazon Transcribe transcription job.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def get_job(job_name, transcribe_client):
    """
    Gets details about a transcription job.

    :param job_name: The name of the job to retrieve.
    :param transcribe_client: The Boto3 Transcribe client.
```

```
:return: The retrieved transcription job.  
"""  
try:  
    response = transcribe_client.get_transcription_job(  
        TranscriptionJobName=job_name)  
    job = response['TranscriptionJob']  
    logger.info("Got job %s.", job['TranscriptionJobName'])  
except ClientError:  
    logger.exception("Couldn't get job %s.", job_name)  
    raise  
else:  
    return job
```

- For API details, see [GetTranscriptionJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## List custom vocabularies

The following code example shows how to list custom Amazon Transcribe vocabularies.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_vocabularies(vocabulary_filter, transcribe_client):  
    """  
    Lists the custom vocabularies created for this AWS account.  
  
    :param vocabulary_filter: The returned vocabularies must contain this string in  
                            their names.  
    :param transcribe_client: The Boto3 Transcribe client.  
    :return: The list of retrieved vocabularies.  
    """  
    try:  
        response = transcribe_client.list_vocabularies(  
            NameContains=vocabulary_filter)  
        vocabs = response['Vocabularies']  
        next_token = response.get('NextToken')  
        while next_token is not None:  
            response = transcribe_client.list_vocabularies(  
                NameContains=vocabulary_filter, NextToken=next_token)  
            vocabs += response['Vocabularies']  
            next_token = response.get('NextToken')  
        logger.info(  
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter)  
    except ClientError:  
        logger.exception(  
            "Couldn't list vocabularies with filter %s.", vocabulary_filter)  
        raise  
    else:  
        return vocabs
```

- For API details, see [ListVocabularies](#) in *AWS SDK for Python (Boto3) API Reference*.

## List transcription jobs

The following code example shows how to list Amazon Transcribe transcription jobs.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def list_jobs(job_filter, transcribe_client):
    """
    Lists summaries of the transcription jobs for the current AWS account.

    :param job_filter: The list of returned jobs must contain this string in their
                       names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved transcription job summaries.
    """

    try:
        response = transcribe_client.list_transcription_jobs(
            JobNameContains=job_filter)
        jobs = response['TranscriptionJobSummaries']
        next_token = response.get('NextToken')
        while next_token is not None:
            response = transcribe_client.list_transcription_jobs(
                JobNameContains=job_filter, NextToken=next_token)
            jobs += response['TranscriptionJobSummaries']
            next_token = response.get('NextToken')
        logger.info("Got %s jobs with filter %s.", len(jobs), job_filter)
    except ClientError:
        logger.exception("Couldn't get jobs with filter %s.", job_filter)
        raise
    else:
        return jobs
```

- For API details, see [ListTranscriptionJobs](#) in *AWS SDK for Python (Boto3) API Reference*.

## Start a transcription job

The following code example shows how to start an Amazon Transcribe transcription job.

## SDK for Python (Boto3)

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def start_job(
    job_name, media_uri, media_format, language_code, transcribe_client,
    vocabulary_name=None):
    """
    Starts a transcription job. This function returns as soon as the job is started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.

    :param job_name: The name of the transcription job. This must be unique for
                     your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
                      in an Amazon S3 bucket.
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
                          For example, en-US or ja-JP
```

```
:param transcribe_client: The Boto3 Transcribe client.
:param vocabulary_name: The name of a custom vocabulary to use when transcribing
                        the audio file.
:return: Data about the job.
"""
try:
    job_args = {
        'TranscriptionJobName': job_name,
        'Media': {'MediaFileUri': media_uri},
        'MediaFormat': media_format,
        'LanguageCode': language_code}
    if vocabulary_name is not None:
        job_args['Settings'] = {'VocabularyName': vocabulary_name}
    response = transcribe_client.start_transcription_job(**job_args)
    job = response['TranscriptionJob']
    logger.info("Started transcription job %s.", job_name)
except ClientError:
    logger.exception("Couldn't start transcription job %s.", job_name)
    raise
else:
    return job
```

- For API details, see [StartTranscriptionJob](#) in *AWS SDK for Python (Boto3) API Reference*.

## Update a custom vocabulary

The following code example shows how to update a custom Amazon Transcribe vocabulary.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
def update_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
    table_uri=None):
"""
Updates an existing custom vocabulary. The entire vocabulary is replaced with
the contents of the update.

:param vocabulary_name: The name of the vocabulary to update.
:param language_code: The language code of the vocabulary.
:param transcribe_client: The Boto3 Transcribe client.
:param phrases: A list of comma-separated phrases to include in the vocabulary.
:param table_uri: A table of phrases and pronunciation hints to include in the
                  vocabulary.
"""
try:
    vocab_args = {'VocabularyName': vocabulary_name, 'LanguageCode': language_code}
    if phrases is not None:
        vocab_args['Phrases'] = phrases
    elif table_uri is not None:
        vocab_args['VocabularyFileUri'] = table_uri
    response = transcribe_client.update_vocabulary(**vocab_args)
    logger.info(
        "Updated custom vocabulary %s.", response['VocabularyName'])
except ClientError:
    logger.exception("Couldn't update custom vocabulary %s.", vocabulary_name)
    raise
```

- For API details, see [UpdateVocabulary](#) in *AWS SDK for Python (Boto3) API Reference*.

## Scenarios

### Create and refine a custom vocabulary

The following code example shows how to:

- Upload an audio file to Amazon S3.
- Run an Amazon Transcribe job to transcribe the file and get the results.
- Create and refine a custom vocabulary to improve transcription accuracy.
- Run jobs with custom vocabularies and get the results.

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Transcribe an audio file that contains a reading of Jabberwocky by Lewis Carroll. Start by creating functions that wrap Amazon Transcribe actions.

```
def start_job(
    job_name, media_uri, media_format, language_code, transcribe_client,
    vocabulary_name=None):
    """
    Starts a transcription job. This function returns as soon as the job is started.
    To get the current status of the job, call get_transcription_job. The job is
    successfully completed when the job status is 'COMPLETED'.

    :param job_name: The name of the transcription job. This must be unique for
        your AWS account.
    :param media_uri: The URI where the audio file is stored. This is typically
        in an Amazon S3 bucket.
    :param media_format: The format of the audio file. For example, mp3 or wav.
    :param language_code: The language code of the audio file.
        For example, en-US or ja-JP
    :param transcribe_client: The Boto3 Transcribe client.
    :param vocabulary_name: The name of a custom vocabulary to use when transcribing
        the audio file.
    :return: Data about the job.
    """
    try:
        job_args = {
            'TranscriptionJobName': job_name,
            'Media': {'MediaFileUri': media_uri},
            'MediaFormat': media_format,
            'LanguageCode': language_code}
        if vocabulary_name is not None:
            job_args['Settings'] = {'VocabularyName': vocabulary_name}
        response = transcribe_client.start_transcription_job(**job_args)
        job = response['TranscriptionJob']
        logger.info("Started transcription job %s.", job_name)
    except ClientError:
        logger.exception("Couldn't start transcription job %s.", job_name)
        raise
    else:
        return job

def get_job(job_name, transcribe_client):
    """
```

```
Gets details about a transcription job.

:param job_name: The name of the job to retrieve.
:param transcribe_client: The Boto3 Transcribe client.
:return: The retrieved transcription job.
"""
try:
    response = transcribe_client.get_transcription_job(
        TranscriptionJobName=job_name)
    job = response['TranscriptionJob']
    logger.info("Got job %s.", job['TranscriptionJobName'])
except ClientError:
    logger.exception("Couldn't get job %s.", job_name)
    raise
else:
    return job

def delete_job(job_name, transcribe_client):
"""
Deletes a transcription job. This also deletes the transcript associated with
the job.

:param job_name: The name of the job to delete.
:param transcribe_client: The Boto3 Transcribe client.
"""
try:
    transcribe_client.delete_transcription_job(
        TranscriptionJobName=job_name)
    logger.info("Deleted job %s.", job_name)
except ClientError:
    logger.exception("Couldn't delete job %s.", job_name)
    raise

def create_vocabulary(
    vocabulary_name, language_code, transcribe_client,
    phrases=None, table_uri=None):
"""
Creates a custom vocabulary that can be used to improve the accuracy of
transcription jobs. This function returns as soon as the vocabulary processing
is started. Call get_vocabulary to get the current status of the vocabulary.
The vocabulary is ready to use when its status is 'READY'.

:param vocabulary_name: The name of the custom vocabulary.
:param language_code: The language code of the vocabulary.
    For example, en-US or nl-NL.
:param transcribe_client: The Boto3 Transcribe client.
:param phrases: A list of comma-separated phrases to include in the vocabulary.
:param table_uri: A table of phrases and pronunciation hints to include in the
    vocabulary.
:return: Information about the newly created vocabulary.
"""
try:
    vocab_args = {'VocabularyName': vocabulary_name, 'LanguageCode': language_code}
    if phrases is not None:
        vocab_args['Phrases'] = phrases
    elif table_uri is not None:
        vocab_args['VocabularyFileUri'] = table_uri
    response = transcribe_client.create_vocabulary(**vocab_args)
    logger.info("Created custom vocabulary %s.", response['VocabularyName'])
except ClientError:
    logger.exception("Couldn't create custom vocabulary %s.", vocabulary_name)
    raise
else:
    return response

def get_vocabulary(vocabulary_name, transcribe_client):
```

```
"""
Gets information about a custom vocabulary.

:param vocabulary_name: The name of the vocabulary to retrieve.
:param transcribe_client: The Boto3 Transcribe client.
:return: Information about the vocabulary.
"""
try:
    response = transcribe_client.get_vocabulary(VocabularyName=vocabulary_name)
    logger.info("Got vocabulary %s.", response['VocabularyName'])
except ClientError:
    logger.exception("Couldn't get vocabulary %s.", vocabulary_name)
    raise
else:
    return response

def update_vocabulary(
    vocabulary_name, language_code, transcribe_client, phrases=None,
    table_uri=None):
    """
    Updates an existing custom vocabulary. The entire vocabulary is replaced with
    the contents of the update.

    :param vocabulary_name: The name of the vocabulary to update.
    :param language_code: The language code of the vocabulary.
    :param transcribe_client: The Boto3 Transcribe client.
    :param phrases: A list of comma-separated phrases to include in the vocabulary.
    :param table_uri: A table of phrases and pronunciation hints to include in the
                      vocabulary.
    """
    try:
        vocab_args = {'VocabularyName': vocabulary_name, 'LanguageCode': language_code}
        if phrases is not None:
            vocab_args['Phrases'] = phrases
        elif table_uri is not None:
            vocab_args['VocabularyFileUri'] = table_uri
        response = transcribe_client.update_vocabulary(**vocab_args)
        logger.info(
            "Updated custom vocabulary %s.", response['VocabularyName'])
    except ClientError:
        logger.exception("Couldn't update custom vocabulary %s.", vocabulary_name)
        raise

def list_vocabularies(vocabulary_filter, transcribe_client):
    """
    Lists the custom vocabularies created for this AWS account.

    :param vocabulary_filter: The returned vocabularies must contain this string in
                             their names.
    :param transcribe_client: The Boto3 Transcribe client.
    :return: The list of retrieved vocabularies.
    """
    try:
        response = transcribe_client.list_vocabularies(
            NameContains=vocabulary_filter)
        vocabs = response['Vocabularies']
        next_token = response.get('NextToken')
        while next_token is not None:
            response = transcribe_client.list_vocabularies(
                NameContains=vocabulary_filter, NextToken=next_token)
            vocabs += response['Vocabularies']
            next_token = response.get('NextToken')
        logger.info(
            "Got %s vocabularies with filter %s.", len(vocabs), vocabulary_filter)
    except ClientError:
        logger.exception(
```

```

        "Couldn't list vocabularies with filter %s.", vocabulary_filter)
    raise
else:
    return vocabs

def delete_vocabulary(vocabulary_name, transcribe_client):
    """
    Deletes a custom vocabulary.

    :param vocabulary_name: The name of the vocabulary to delete.
    :param transcribe_client: The Boto3 Transcribe client.
    """
    try:
        transcribe_client.delete_vocabulary(VocabularyName=vocabulary_name)
        logger.info("Deleted vocabulary %s.", vocabulary_name)
    except ClientError:
        logger.exception("Couldn't delete vocabulary %s.", vocabulary_name)
        raise

```

Call the wrapper functions to transcribe audio without a custom vocabulary and then with different versions of a custom vocabulary to see improved results.

```

def usage_demo():
    """Shows how to use the Amazon Transcribe service."""
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    s3_resource = boto3.resource('s3')
    transcribe_client = boto3.client('transcribe')

    print('*'*88)
    print("Welcome to the Amazon Transcribe demo!")
    print('*'*88)

    bucket_name = f'jabber-bucket-{time.time_ns()}''
    print(f"Creating bucket {bucket_name}.")
    bucket = s3_resource.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            'LocationConstraint': transcribe_client.meta.region_name})
    media_file_name = '.media/Jabberwocky.mp3'
    media_object_key = 'Jabberwocky.mp3'
    print(f"Uploading media file {media_file_name}.")
    bucket.upload_file(media_file_name, media_object_key)
    media_uri = f's3://{bucket.name}/{media_object_key}'

    job_name_simple = f'Jabber-{time.time_ns()}''
    print(f"Starting transcription job {job_name_simple}.")
    start_job(
        job_name_simple, f's3://{bucket_name}/{media_object_key}', 'mp3', 'en-US',
        transcribe_client)
    transcribe_waiter = TranscribeCompleteWaiter(transcribe_client)
    transcribe_waiter.wait(job_name_simple)
    job_simple = get_job(job_name_simple, transcribe_client)
    transcript_simple = requests.get(
        job_simple['Transcript']['TranscriptFileUri']).json()
    print(f"Transcript for job {transcript_simple['jobName']}:")
    print(transcript_simple['results'][0]['transcripts'][0]['transcript'])

    print('*'*88)
    print("Creating a custom vocabulary that lists the nonsense words to try to "
          "improve the transcription.")
    vocabulary_name = f'Jabber-vocabulary-{time.time_ns()}''
    create_vocabulary(

```

```

vocabulary_name, 'en-US', transcribe_client,
phrases=[

    'brillig', 'slithy', 'borogoves', 'mome', 'raths', 'Jub-Jub', 'frumious',
    'manxome', 'Tumtum', 'uffish', 'whiffling', 'tulgey', 'thou', 'frabjous',
    'callooh', 'callay', 'chortled'],
)
vocabulary_ready_waiter = VocabularyReadyWaiter(transcribe_client)
vocabulary_ready_waiter.wait(vocabulary_name)

job_name_vocabulary_list = f'Jabber-vocabulary-list-{time.time_ns()}' 
print(f"Starting transcription job {job_name_vocabulary_list}.") 
start_job(
    job_name_vocabulary_list, media_uri, 'mp3', 'en-US', transcribe_client,
    vocabulary_name)
transcribe_waiter.wait(job_name_vocabulary_list)
job_vocabulary_list = get_job(job_name_vocabulary_list, transcribe_client)
transcript_vocabulary_list = requests.get(
    job_vocabulary_list['Transcript']['TranscriptFileUri']).json()
print(f"Transcript for job {transcript_vocabulary_list['jobName']}:")
print(transcript_vocabulary_list['results'][0]['transcripts'][0]['transcript'])

print('*'*88)
print("Updating the custom vocabulary with table data that provides additional "
      "pronunciation hints.")
table_vocab_file = 'jabber-vocabulary-table.txt'
bucket.upload_file(table_vocab_file, table_vocab_file)
update_vocabulary(
    vocabulary_name, 'en-US', transcribe_client,
    table_uri=f's3://{bucket.name}/{table_vocab_file}')
vocabulary_ready_waiter.wait(vocabulary_name)

job_name_vocab_table = f'Jabber-vocab-table-{time.time_ns()}' 
print(f"Starting transcription job {job_name_vocab_table}.") 
start_job(
    job_name_vocab_table, media_uri, 'mp3', 'en-US', transcribe_client,
    vocabulary_name=vocabulary_name)
transcribe_waiter.wait(job_name_vocab_table)
job_vocab_table = get_job(job_name_vocab_table, transcribe_client)
transcript_vocab_table = requests.get(
    job_vocab_table['Transcript']['TranscriptFileUri']).json()
print(f"Transcript for job {transcript_vocab_table['jobName']}:")
print(transcript_vocab_table['results'][0]['transcripts'][0]['transcript'])

print('*'*88)
print("Getting data for jobs and vocabularies.")
jabber_jobs = list_jobs('Jabber', transcribe_client)
print(f"Found {len(jabber_jobs)} jobs:")
for job_sum in jabber_jobs:
    job = get_job(job_sum['TranscriptionJobName'], transcribe_client)
    print(f"\t{job['TranscriptionJobName']}, {job['Media']['MediaFileUri']}, "
          f"{job['Settings'].get('VocabularyName')}")

jabber_vocabs = list_vocabularies('Jabber', transcribe_client)
print(f"Found {len(jabber_vocabs)} vocabularies:")
for vocab_sum in jabber_vocabs:
    vocab = get_vocabulary(vocab_sum['VocabularyName'], transcribe_client)
    vocab_content = requests.get(vocab['DownloadUri']).text
    print(f"\t{vocab['VocabularyName']} contents:")
    print(vocab_content)

print('*'*88)
print("Deleting demo jobs.")
for job_name in [job_name_simple, job_name_vocabulary_list, job_name_vocab_table]:
    delete_job(job_name, transcribe_client)
print("Deleting demo vocabulary.")
delete_vocabulary(vocabulary_name, transcribe_client)

```

```
print("Deleting demo bucket.")
bucket.objects.delete()
bucket.delete()
print("Thanks for watching!")
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [CreateVocabulary](#)
  - [DeleteTranscriptionJob](#)
  - [DeleteVocabulary](#)
  - [GetTranscriptionJob](#)
  - [GetVocabulary](#)
  - [ListVocabularies](#)
  - [StartTranscriptionJob](#)
  - [UpdateVocabulary](#)

## Transcribe audio and get job data

The following code example shows how to:

- Start a transcription job with Amazon Transcribe.
- Wait for the job to complete.
- Get the URI where the transcript is stored.

For more information, see [Getting started with Amazon Transcribe](#).

### SDK for Python (Boto3)

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
import time
import boto3

def transcribe_file(job_name, file_uri, transcribe_client):
    transcribe_client.start_transcription_job(
        TranscriptionJobName=job_name,
        Media={'MediaFileUri': file_uri},
        MediaFormat='wav',
        LanguageCode='en-US'
    )

    max_tries = 60
    while max_tries > 0:
        max_tries -= 1
        job = transcribe_client.get_transcription_job(TranscriptionJobName=job_name)
        job_status = job['TranscriptionJob']['TranscriptionJobStatus']
        if job_status in ['COMPLETED', 'FAILED']:
            print(f"Job {job_name} is {job_status}.")
            if job_status == 'COMPLETED':
                print(
                    f"Download the transcript from\n"
                    f"\t{job['TranscriptionJob']['Transcript']['TranscriptFileUri']}.")

        break
    else:
```

```
        print(f"Waiting for {job_name}. Current status is {job_status}.")  
        time.sleep(10)  
  
def main():  
    transcribe_client = boto3.client('transcribe')  
    file_uri = 's3://test-transcribe/answer2.wav'  
    transcribe_file('Example-job', file_uri, transcribe_client)  
  
if __name__ == '__main__':  
    main()
```

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - [GetTranscriptionJob](#)
  - [StartTranscriptionJob](#)

## Cross-service examples using SDK for Python (Boto3)

The following sample applications use the AWS SDK for Python (Boto3) to work across multiple AWS services.

### Examples

- [Create an API Gateway REST API to track COVID-19 data \(p. 4304\)](#)
- [Create a lending library REST API \(p. 4305\)](#)
- [Create a long-lived Amazon EMR cluster and run several steps using an AWS SDK \(p. 4305\)](#)
- [Create a messenger application with Step Functions \(p. 4306\)](#)
- [Create a short-lived Amazon EMR cluster and run a step using an AWS SDK \(p. 4306\)](#)
- [Create a web application to track DynamoDB data \(p. 4307\)](#)
- [Create a websocket chat application with API Gateway \(p. 4307\)](#)
- [Create an Aurora Serverless work item tracker \(p. 4307\)](#)
- [Create an Amazon Textract explorer application \(p. 4308\)](#)
- [Detect entities in text extracted from an image using an AWS SDK \(p. 4308\)](#)
- [Detect objects in images with Amazon Rekognition using an AWS SDK \(p. 4309\)](#)
- [Detect people and objects in a video with Amazon Rekognition using an AWS SDK \(p. 4309\)](#)
- [Use API Gateway to invoke a Lambda function \(p. 4310\)](#)
- [Use scheduled events to invoke a Lambda function \(p. 4310\)](#)

## Create an API Gateway REST API to track COVID-19 data

### SDK for Python (Boto3)

Shows how to use AWS Chalice with the AWS SDK for Python (Boto3) to create a serverless REST API that uses Amazon API Gateway, AWS Lambda, and Amazon DynamoDB. The REST API simulates a system that tracks daily cases of COVID-19 in the United States, using fictional data. Learn how to:

- Use AWS Chalice to define routes in Lambda functions that are called to handle REST requests that come through API Gateway.
- Use Lambda functions to retrieve and store data in a DynamoDB table to serve REST requests.
- Define table structure and security role resources in an AWS CloudFormation template.
- Use AWS Chalice and CloudFormation to package and deploy all necessary resources.

- Use CloudFormation to clean up all created resources.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- API Gateway
- AWS CloudFormation
- DynamoDB
- Lambda

## Create a lending library REST API

#### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with the Amazon Relational Database Service (Amazon RDS) API and AWS Chalice to create a REST API backed by an Amazon Aurora database. The web service is fully serverless and represents a simple lending library where patrons can borrow and return books. Learn how to:

- Create and manage a serverless Aurora database cluster.
- Use AWS Secrets Manager to manage database credentials.
- Implement a data storage layer that uses Amazon RDS to move data into and out of the database.
- Use AWS Chalice to deploy a serverless REST API to Amazon API Gateway and AWS Lambda.
- Use the Requests package to send requests to the web service.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- API Gateway
- Lambda
- Amazon RDS
- Secrets Manager

## Create a long-lived Amazon EMR cluster and run several steps using an AWS SDK

#### SDK for Python (Boto3)

Create a long-lived Amazon EMR cluster that uses Apache Spark to query historical Amazon review data from the [Amazon Customer Reviews Dataset](#). Run a job that gets data for top-rated products in specific categories that contain keywords in their product titles. Job results are written to an Amazon Simple Storage Service (Amazon S3) bucket.

- Create an Amazon S3 bucket and upload a job script.
- Create AWS Identity and Access Management (IAM) roles.
- Create Amazon Elastic Compute Cloud (Amazon EC2) security groups.
- Create a long-lived cluster and run several job steps.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

#### Services used in this example

- Amazon EC2

- Amazon EMR
- IAM
- Amazon S3

## Create a messenger application with Step Functions

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with AWS Step Functions to create a messenger application that retrieves message records from an Amazon DynamoDB table and sends them with Amazon Simple Queue Service (Amazon SQS). The state machine integrates with an AWS Lambda function to scan the database for unsent messages.

- Create a state machine that retrieves and updates message records from an Amazon DynamoDB table.
- Update the state machine definition to also send messages to Amazon Simple Queue Service (Amazon SQS).
- Start and stop state machine runs.
- Connect to Lambda, DynamoDB, and Amazon SQS from a state machine by using service integrations.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Lambda
- Amazon SQS
- Step Functions

## Create a short-lived Amazon EMR cluster and run a step using an AWS SDK

### SDK for Python (Boto3)

Create a short-lived Amazon EMR cluster that estimates the value of pi using Apache Spark to parallelize a large number of calculations. The job writes output to Amazon EMR logs and to an Amazon Simple Storage Service (Amazon S3) bucket. The cluster terminates itself after completing the job.

- Create an Amazon S3 bucket and upload a job script.
- Create AWS Identity and Access Management (IAM) roles.
- Create Amazon Elastic Compute Cloud (Amazon EC2) security groups.
- Create a short-lived cluster and run a single job step.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon EC2
- Amazon EMR
- IAM
- Amazon S3

## Create a web application to track DynamoDB data

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) to create a REST service that tracks work items in Amazon DynamoDB and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses the Flask web framework to handle HTTP routing and integrates with a React webpage to present a fully functional web application.

- Build a Flask REST service that integrates with AWS services.
- Read, write, and update work items that are stored in a DynamoDB table.
- Use Amazon SES to send email reports of work items.

For complete source code and instructions on how to set up and run, see the full example in the [AWS Code Examples Repository](#) on GitHub.

### Services used in this example

- DynamoDB
- Amazon SES

## Create a websocket chat application with API Gateway

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with Amazon API Gateway V2 to create a websocket API that integrates with AWS Lambda and Amazon DynamoDB.

- Create a websocket API served by API Gateway.
- Define a Lambda handler that stores connections in DynamoDB and posts messages to other chat participants.
- Connect to the websocket chat application and send messages with the Websockets package.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- API Gateway
- DynamoDB
- Lambda

## Create an Aurora Serverless work item tracker

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) to create a REST service that tracks work items in an Amazon Aurora Serverless database and emails reports by using Amazon Simple Email Service (Amazon SES). This example uses the Flask web framework to handle HTTP routing and integrates with a React webpage to present a fully functional web application.

- Build a Flask REST service that integrates with AWS services.
- Read, write, and update work items that are stored in an Aurora Serverless database.
- Create an AWS Secrets Manager secret that contains database credentials and use it to authenticate calls to the database.
- Use Amazon SES to send email reports of work items.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

## Create an Amazon Textract explorer application

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) with Amazon Textract to detect text, form, and table elements in a document image. The input image and Amazon Textract output are shown in a Tkinter application that lets you explore the detected elements.

- Submit a document image to Amazon Textract and explore the output of detected elements.
- Submit images directly to Amazon Textract or through an Amazon Simple Storage Service (Amazon S3) bucket.
- Use asynchronous APIs to start a job that publishes a notification to an Amazon Simple Notification Service (Amazon SNS) topic when the job completes.
- Poll an Amazon Simple Queue Service (Amazon SQS) queue for a job completion message and display the results.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

## Detect entities in text extracted from an image using an AWS SDK

### SDK for Python (Boto3)

Shows how to use the AWS SDK for Python (Boto3) in a Jupyter notebook to detect entities in text that is extracted from an image. This example uses Amazon Textract to extract text from an image stored in Amazon Simple Storage Service (Amazon S3) and Amazon Comprehend to detect entities in the extracted text.

This example is a Jupyter notebook and must be run in an environment that can host notebooks. For instructions on how to run the example using Amazon SageMaker, see the directions in [ExtractAndComprehendNotebook.ipynb](#).

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Comprehend
- Amazon S3
- Amazon Textract

## Detect objects in images with Amazon Rekognition using an AWS SDK

### SDK for Python (Boto3)

Shows you how to use the AWS SDK for Python (Boto3) to create a web application that lets you do the following:

- Upload photos to an Amazon Simple Storage Service (Amazon S3) bucket.
- Use Amazon Rekognition to analyze and label the photos.
- Use Amazon Simple Email Service (Amazon SES) to send email reports of image analysis.

This example contains two main components: a webpage written in JavaScript that is built with React, and a REST service written in Python that is built with Flask-RESTful.

You can use the React webpage to:

- Display a list of images that are stored in your S3 bucket.
- Upload images from your computer to your S3 bucket.
- Display images and labels that identify items that are detected in the image.
- Get a report of all images in your S3 bucket and send an email of the report.

The webpage calls the REST service. The service sends requests to AWS to perform the following actions:

- Get and filter the list of images in your S3 bucket.
- Upload photos to your S3 bucket.
- Use Amazon Rekognition to analyze individual photos and get a list of labels that identify items that are detected in the photo.
- Analyze all photos in your S3 bucket and use Amazon SES to email a report.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon S3
- Amazon SES

## Detect people and objects in a video with Amazon Rekognition using an AWS SDK

### SDK for Python (Boto3)

Use Amazon Rekognition to detect faces, objects, and people in videos by starting asynchronous detection jobs. This example also configures Amazon Rekognition to notify an Amazon Simple Notification Service (Amazon SNS) topic when jobs complete and subscribes an Amazon Simple Queue Service (Amazon SQS) queue to the topic. When the queue receives a message about a job, the job is retrieved and the results are output.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon SNS

- Amazon SQS

## Use API Gateway to invoke a Lambda function

### SDK for Python (Boto3)

This example shows how to create and use an Amazon API Gateway REST API that targets an AWS Lambda function. The Lambda handler demonstrates how to route based on HTTP methods; how to get data from the query string, header, and body; and how to return a JSON response.

- Deploy a Lambda function.
- Create an API Gateway REST API.
- Create a REST resource that targets the Lambda function.
- Grant permission to let API Gateway invoke the Lambda function.
- Use the Requests package to send requests to the REST API.
- Clean up all resources created during the demo.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- API Gateway
- Lambda

## Use scheduled events to invoke a Lambda function

### SDK for Python (Boto3)

This example shows how to register an AWS Lambda function as the target of a scheduled Amazon EventBridge event. The Lambda handler writes a friendly message and the full event data to Amazon CloudWatch Logs for later retrieval.

- Deploys a Lambda function.
- Creates an EventBridge scheduled event and makes the Lambda function the target.
- Grants permission to let EventBridge invoke the Lambda function.
- Prints the latest data from CloudWatch Logs to show the result of the scheduled invocations.
- Cleans up all resources created during the demo.

This example is best viewed on GitHub. For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- CloudWatch Logs
- EventBridge
- Lambda

## Code examples for SDK for Ruby

The code examples in this topic show you how to use the AWS SDK for Ruby with AWS.

### Examples

- [Single-service actions and scenarios using SDK for Ruby \(p. 4311\)](#)

# Single-service actions and scenarios using SDK for Ruby

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for Ruby with AWS services.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Services

- [DynamoDB examples using SDK for Ruby \(p. 4311\)](#)
- [EventBridge examples using SDK for Ruby \(p. 4324\)](#)
- [IAM examples using SDK for Ruby \(p. 4338\)](#)
- [Lambda examples using SDK for Ruby \(p. 4356\)](#)
- [Amazon S3 examples using SDK for Ruby \(p. 4366\)](#)
- [Amazon SNS examples using SDK for Ruby \(p. 4385\)](#)
- [AWS STS examples using SDK for Ruby \(p. 4390\)](#)

## DynamoDB examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Amazon DynamoDB.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Topics

- [Actions \(p. 4311\)](#)
- [Scenarios \(p. 4317\)](#)

## Actions

### Create a table

The following code example shows how to create a DynamoDB table.

#### SDK for Ruby

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates an Amazon DynamoDB table that can be used to store movie data.  
# The table uses the release year of the movie as the partition key and the  
# title as the sort key.  
#  
# @param table_name [String] The name of the table to create.  
# @return [Aws::Dynamodb::Table] The newly created table.  
def create_table(table_name)  
    @table = @dynamo_resource.create_table(  
        :table_name => table_name,  
        :attribute_definitions => [
```

```
table_name: table_name,
key_schema: [
    {attribute_name: "year", key_type: "HASH"}, # Partition key
    {attribute_name: "title", key_type: "RANGE"} # Sort key
],
attribute_definitions: [
    {attribute_name: "year", attribute_type: "N"},
    {attribute_name: "title", attribute_type: "S"}
],
provisioned_throughput: {read_capacity_units: 10, write_capacity_units: 10})
@dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
rescue Aws::Errors::ServiceError => e
  puts("Couldn't create table #{table_name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  @table
end
```

- For API details, see [CreateTable](#) in *AWS SDK for Ruby API Reference*.

## Delete a table

The following code example shows how to delete a DynamoDB table.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes the table.
def delete_table
  @table.delete
  @table = nil
rescue Aws::Errors::ServiceError => e
  puts("Couldn't delete table. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- For API details, see [DeleteTable](#) in *AWS SDK for Ruby API Reference*.

## Delete an item from a table

The following code example shows how to delete an item from a DynamoDB table.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a movie from the table.
#
# @param title [String] The title of the movie to delete.
# @param year [Integer] The release year of the movie to delete.
def delete_movie(title, year)
```

```
    @table.delete_item(key: {"year" => year, "title" => title})
rescue Aws::Errors::ServiceError => e
  puts("Couldn't delete movie #{title}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- For API details, see [DeleteItem](#) in *AWS SDK for Ruby API Reference*.

## Get an item from a table

The following code example shows how to get an item from a DynamoDB table.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Gets movie data from the table for a specific movie.
#
# @param title [String] The title of the movie.
# @param year [Integer] The release year of the movie.
# @return [Hash] The data about the requested movie.
def get_movie(title, year)
  response = @table.get_item(key: {"year" => year, "title" => title})
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get movie #{title} from table #{@table.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  response.item
end
```

- For API details, see [GetItem](#) in *AWS SDK for Ruby API Reference*.

## Get information about a table

The following code example shows how to get information about a DynamoDB table.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Determines whether a table exists. As a side effect, stores the table in
# a member variable.
#
# @param table_name [String] The name of the table to check.
# @return [Boolean] True when the table exists; otherwise, False.
def exists?(table_name)
  table = Aws::DynamoDB::Table.new(table_name)
  table.load
  @table = table
rescue Aws::DynamoDB::Errors::ResourceNotFoundException
  puts("Table #{table_name} doesn't exist. Let's create it.")
```

```
    false
rescue Aws::Errors::ServiceError => e
  puts("Couldn't check for existence of #{table_name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  !@table.nil?
end
```

- For API details, see [DescribeTable](#) in *AWS SDK for Ruby API Reference*.

## Put an item in a table

The following code example shows how to put an item in a DynamoDB table.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Adds a movie to the table.
#
# @param title [String] The title of the movie.
# @param year [Integer] The release year of the movie.
# @param plot [String] The plot summary of the movie.
# @param rating [Float] The quality rating of the movie.
def add_movie(title:, year:, plot:, rating:)
  @table.put_item(
    item: {
      "year" => year,
      "title" => title,
      "info" => {"plot" => plot, "rating" => rating}})
rescue Aws::Errors::ServiceError => e
  puts("Couldn't add movie #{title} to table #{@table.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- For API details, see [PutItem](#) in *AWS SDK for Ruby API Reference*.

## Query a table

The following code example shows how to query a DynamoDB table.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Queries for movies that were released in the specified year.
#
# @param year [Integer] The year to query.
# @return [Array] The list of movies that were released in the specified year.
def query_movies(year)
  response = @table.query(
```

```
key_condition_expression: "#yr = :year",
expression_attribute_names: {"#yr" => "year"},
expression_attribute_values: {":year" => year})
rescue Aws::Errors::ServiceError => e
  puts("Couldn't query for movies released in #{year}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  response.items
end
```

- For API details, see [Query in AWS SDK for Ruby API Reference](#).

## Scan a table

The following code example shows how to scan a DynamoDB table.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Scans for movies that were released in a range of years.
# Uses a projection expression to return a subset of data for each movie.
#
# @param year_range [Hash] The range of years to retrieve.
# @return [Array] The list of movies released in the specified years.
def scan_movies(year_range)
  movies = []
  scan_hash = {
    filter_expression: "#yr between :start_yr and :end_yr",
    projection_expression: "#yr, title, info.rating",
    expression_attribute_names: {"#yr" => "year"},
    expression_attribute_values: {
      ":start_yr" => year_range[:start], ":end_yr" => year_range[:end]}
  }
  done = false
  start_key = nil
  until done
    scan_hash[:exclusive_start_key] = start_key unless start_key.nil?
    response = @table.scan(scan_hash)
    movies.concat(response.items) unless response.items.nil?
    start_key = response.last_evaluated_key
    done = start_key.nil?
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't scan for movies. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  movies
end
```

- For API details, see [Scan in AWS SDK for Ruby API Reference](#).

## Update an item in a table

The following code example shows how to update an item in a DynamoDB table.

## SDK for Ruby

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Updates rating and plot data for a movie in the table.  
#  
# @param title [String] The title of the movie to update.  
# @param year [Int] The release year of the movie to update.  
# @param rating [Float] The updated rating to give the movie.  
# @param plot [String] The updated plot summary to give the movie.  
# @return [Hash] The fields that were updated, with their new values.  
def update_movie(title:, year:, rating:, plot:)  
    response = @table.update_item(  
        key: {"year" => year, "title" => title},  
        update_expression: "set info.rating=:r, info.plot=:p",  
        expression_attribute_values: { ":r" => rating, ":p" => plot },  
        return_values: "UPDATED_NEW")  
    rescue Aws::Errors::ServiceError => e  
        puts("Couldn't update movie #{title} in table #{@table.name}. Here's why:")  
        puts("\t#{e.code}: #{e.message}")  
        raise  
    else  
        response.attributes  
    end
```

- For API details, see [UpdateItem](#) in *AWS SDK for Ruby API Reference*.

## Write a batch of items

The following code example shows how to write a batch of DynamoDB items.

## SDK for Ruby

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Fills an Amazon DynamoDB table with the specified data. Items are sent in  
# batches of 25 until all items are written.  
#  
# @param movies [Enumerable] The data to put in the table. Each item must contain at  
# least the keys required by the schema that was specified when  
# the table was created.  
def write_batch(movies)  
    index = 0  
    slice_size = 25  
    while index < movies.length  
        movie_items = []  
        movies[index, slice_size].each do |movie|  
            movie_items.append({put_request: { item: movie }})  
        end  
        @dynamo_resource.batch_write_item({request_items: { @table.name =>  
            movie_items }})  
        index += slice_size  
    end
```

```
rescue Aws::Errors::ServiceError => e
  puts(
    "Couldn't load data into table #{@table.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- For API details, see [BatchWriteItem](#) in *AWS SDK for Ruby API Reference*.

## Scenarios

### Get started using tables, items, and queries

The following code example shows how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.
- Delete a movie from the table.
- Delete the table.

## SDK for Ruby

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that encapsulates a DynamoDB table.

```
require "aws-sdk-dynamodb"
require "json"
require "open-uri"
require "pp"
require "zip"
require_relative "question"

# Encapsulates an Amazon DynamoDB table of movie data.
class Movies
  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(dynamo_resource)
    @dynamo_resource = dynamo_resource
  end

  # Determines whether a table exists. As a side effect, stores the table in
  # a member variable.
  #
  # @param table_name [String] The name of the table to check.
  # @return [Boolean] True when the table exists; otherwise, False.
  def exists?(table_name)
    table = Aws::DynamoDB::Table.new(table_name)
    table.load
    @table = table
  rescue Aws::DynamoDB::Errors::ResourceNotFoundException
```

```
    puts("Table #{table_name} doesn't exist. Let's create it.")
    false
rescue Aws::Errors::ServiceError => e
    puts("Couldn't check for existence of #{table_name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    !@table.nil?
end

# Creates an Amazon DynamoDB table that can be used to store movie data.
# The table uses the release year of the movie as the partition key and the
# title as the sort key.
#
# @param table_name [String] The name of the table to create.
# @return [Aws::DynamoDB::Table] The newly created table.
def create_table(table_name)
    @table = @dynamo_resource.create_table(
        table_name: table_name,
        key_schema: [
            {attribute_name: "year", key_type: "HASH"}, # Partition key
            {attribute_name: "title", key_type: "RANGE"} # Sort key
        ],
        attribute_definitions: [
            {attribute_name: "year", attribute_type: "N"},
            {attribute_name: "title", attribute_type: "S"}
        ],
        provisioned_throughput: {read_capacity_units: 10, write_capacity_units: 10})
    @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
rescue Aws::Errors::ServiceError => e
    puts("Couldn't create table #{table_name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    @table
end

# Fills an Amazon DynamoDB table with the specified data. Items are sent in
# batches of 25 until all items are written.
#
# @param movies [Enumerable] The data to put in the table. Each item must contain at
# least
#                               the keys required by the schema that was specified when
# the
#                               table was created.
def write_batch(movies)
    index = 0
    slice_size = 25
    while index < movies.length
        movie_items = []
        movies[index, slice_size].each do |movie|
            movie_items.append({put_request: { item: movie }})
        end
        @dynamo_resource.batch_write_item({request_items: { @table.name =>
            movie_items }})
        index += slice_size
    end
rescue Aws::Errors::ServiceError => e
    puts(
        "Couldn't load data into table #{@table.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
end

# Adds a movie to the table.
#
```

```
# @param title [String] The title of the movie.
# @param year [Integer] The release year of the movie.
# @param plot [String] The plot summary of the movie.
# @param rating [Float] The quality rating of the movie.
def add_movie(title:, year:, plot:, rating:)
    @table.put_item(
        item: {
            "year" => year,
            "title" => title,
            "info" => {"plot" => plot, "rating" => rating}})
    rescue Aws::Errors::ServiceError => e
        puts("Couldn't add movie #{title} to table #{@table.name}. Here's why:")
        puts("\t#{e.code}: #{e.message}")
        raise
    end

    # Gets movie data from the table for a specific movie.
    #
    # @param title [String] The title of the movie.
    # @param year [Integer] The release year of the movie.
    # @return [Hash] The data about the requested movie.
    def get_movie(title, year)
        response = @table.get_item(key: {"year" => year, "title" => title})
    rescue Aws::Errors::ServiceError => e
        puts("Couldn't get movie #{title} from table #{@table.name}. Here's why:")
        puts("\t#{e.code}: #{e.message}")
        raise
    else
        response.item
    end

    # Updates rating and plot data for a movie in the table.
    #
    # @param title [String] The title of the movie to update.
    # @param year [Int] The release year of the movie to update.
    # @param rating [Float] The updated rating to give the movie.
    # @param plot [String] The updated plot summary to give the movie.
    # @return [Hash] The fields that were updated, with their new values.
    def update_movie(title:, year:, rating:, plot:)
        response = @table.update_item(
            key: {"year" => year, "title" => title},
            update_expression: "set info.rating=:r, info.plot=:p",
            expression_attribute_values: { ":r" => rating, ":p" => plot },
            return_values: "UPDATED_NEW")
    rescue Aws::Errors::ServiceError => e
        puts("Couldn't update movie #{title} in table #{@table.name}. Here's why:")
        puts("\t#{e.code}: #{e.message}")
        raise
    else
        response.attributes
    end

    # Queries for movies that were released in the specified year.
    #
    # @param year [Integer] The year to query.
    # @return [Array] The list of movies that were released in the specified year.
    def query_movies(year)
        response = @table.query(
            key_condition_expression: "#yr = :year",
            expression_attribute_names: {"#yr" => "year"},
            expression_attribute_values: {":year" => year})
    rescue Aws::Errors::ServiceError => e
        puts("Couldn't query for movies released in #{year}. Here's why:")
        puts("\t#{e.code}: #{e.message}")
        raise
    else
```

```

        response.items
    end

    # Scans for movies that were released in a range of years.
    # Uses a projection expression to return a subset of data for each movie.
    #
    # @param year_range [Hash] The range of years to retrieve.
    # @return [Array] The list of movies released in the specified years.
    def scan_movies(year_range)
        movies = []
        scan_hash = {
            filter_expression: "#yr between :start_yr and :end_yr",
            projection_expression: "#yr, title, info.rating",
            expression_attribute_names: {"#yr" => "year"},
            expression_attribute_values: {
                ":start_yr" => year_range[:start], ":end_yr" => year_range[:end]}
        }
        done = false
        start_key = nil
        until done
            scan_hash[:exclusive_start_key] = start_key unless start_key.nil?
            response = @table.scan(scan_hash)
            movies.concat(response.items) unless response.items.nil?
            start_key = response.last_evaluated_key
            done = start_key.nil?
        end
        rescue Aws::Errors::ServiceError => e
            puts("Couldn't scan for movies. Here's why:")
            puts("\t#{e.code}: #{e.message}")
            raise
        else
            movies
        end

        # Deletes a movie from the table.
        #
        # @param title [String] The title of the movie to delete.
        # @param year [Integer] The release year of the movie to delete.
        def delete_movie(title, year)
            @table.delete_item(key: {"year" => year, "title" => title})
        rescue Aws::Errors::ServiceError => e
            puts("Couldn't delete movie #{title}. Here's why:")
            puts("\t#{e.code}: #{e.message}")
            raise
        end

        # Deletes the table.
        def delete_table
            @table.delete
            @table = nil
        rescue Aws::Errors::ServiceError => e
            puts("Couldn't delete table. Here's why:")
            puts("\t#{e.code}: #{e.message}")
            raise
        end
    end

```

Create a helper function to download and extract the sample JSON file.

```

# Gets sample movie data, either from a local file or by first downloading it from
# the Amazon DynamoDB Developer Guide.
#
# @param movie_file_name [String] The local file name where the movie data is stored in
# JSON format.

```

```
# @return [Hash] The movie data as a Hash.
def get_sample_movie_data(movie_file_name)
  if !File.file?(movie_file_name)
    puts("Downloading #{movie_file_name}...")
    movie_content = URI.open(
      "https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/moviedata.zip"
    )
    movie_json = ""
    Zip::File.open_buffer(movie_content) do |zip|
      zip.each do |entry|
        movie_json = entry.get_input_stream.read
      end
    end
  else
    movie_json = File.read(movie_file_name)
  end
  movie_data = JSON.parse(movie_json)
  # The sample file lists over 4000 movies. This returns only the first 250.
  movie_data.slice(0, 250)
rescue Errno::ENOENT
  puts("File #{movie_file_name} not found. Before you can run this demo, you must \"\n    download the file. For instructions, see the README.\")")
  raise
end
```

Run an interactive scenario to create the table and perform actions on it.

```
# Runs the DynamoDB getting started demo.
#
# @param movies [Movies] A wrapper class initialized with a DynamoDB resource.
# @param table_name [String] The name to give the movie table.
# @param movie_file_name [String] The name of a file that contains movie data in JSON
#                               format. This data is loaded into the movie table
#                               as part of the demo.
def run_scenario(movies, table_name, movie_file_name)
  puts("-" * 88)
  puts("Welcome to the DynamoDB getting started demo.")
  puts("-" * 88)

  movies_exists = movies.exists?(table_name)
  unless movies_exists
    puts("\nCreating table #{table_name}...")
    movies.create_table(table_name)
    puts("\nCreated table #{movies.table.name}.")
  end

  my_movie = {}
  my_movie[:title] = Question.ask("Enter the title of a movie to add to the table: ")
  my_movie[:year] = Question.ask("What year was it released? ", method(:is_int))
  my_movie[:rating] = Question.ask(
    "On a scale of 1 - 10, how do you rate it? ", method(:is_float), in_range(1, 10)
  )
  my_movie[:plot] = Question.ask("Summarize the plot for me: ")

  movies.add_movie(**my_movie)
  puts("\nAdded '#{my_movie[:title]}' to '#{movies.table.name}'")
  puts("-" * 88)

  puts("Let's update your movie. You rated it #{my_movie[:rating]}")
  my_movie[:rating] = Question.ask("What new rating would you give it? ",
    method(:is_float), in_range(1, 10))
  puts("You summarized the plot as '#{my_movie[:plot]}'.")
  my_movie[:plot] = Question.ask("What would you say now? ")
```

```
updated = movies.update_movie(**my_movie)
puts("Updated '#{my_movie[:title]}' with new attributes:")
pp(updated)
puts("-" * 88)

unless movies_exists
  movie_data = get_sample_movie_data(movie_file_name)
  puts("Reading data from '#{movie_file_name}' into your table.")
  movies.write_batch(movie_data)
  puts("Wrote #{movie_data.length} movies into #{movies.table.name}.")
  puts("-" * 88)
end

title = "The Lord of the Rings: The Fellowship of the Ring"
if Question.ask("Let's move on. Do you want to get info about '#{title}'? (y/n) ",
  method(:is_yesno))
  movie = movies.get_movie(title, 2001)
  puts("\nHere's what I found:")
  pp(movie)
  puts("-" * 88)
end

ask_for_year = true
puts("Let's get a list of movies released in a given year.")
while ask_for_year
  release_year = Question.ask(
    "Enter a year between 1972 and 2018: ", method(:is_int), in_range(1972, 2018))
  releases = movies.query_movies(release_year)
  if !releases.empty?
    puts("There were #{releases.length} movies released in #{release_year}:")
    releases.each do |release|
      puts("\t#{release["title"]}")
    end
    ask_for_year = false
  else
    puts("I don't know about any movies released in #{release_year}!")
    ask_for_year = Question.ask("Try another year? (y/n) ", method(:is_yesno))
    puts("-" * 88)
  end
end

years = []
years[:start] = Question.ask(
  "Let's scan for movies released in a range of years. Enter a year: ",
  method(:is_int), in_range(1972, 2018))
years[:end] = Question.ask(
  "Now enter another year: ", method(:is_int), in_range(1972, 2018))
releases = movies.scan_movies(years)
if !releases.empty?
  puts("Found #{releases.length} movies.")
  count = Question.ask(
    "How many do you want to see? ", method(:is_int), in_range(1, releases.length))
  puts("Here are your #{count} movies:")
  releases.take(count).each do |release|
    puts("\t#{release["title"]}")
  end
else
  puts("I don't know about any movies released between #{years[:start]} \"\
        \"and #{years[:end]}\"")
  puts("-" * 88)
end

puts("Let's remove your movie from the table.")
if Question.ask(
  "Do you want to remove '#{my_movie[:title]}'? (y/n) ", method(:is_yesno))
  movies.delete_movie(my_movie[:title], my_movie[:year])
end
```

```

    puts("Removed '#{my_movie[:title]}' from the table.")
    puts("-" * 88)
end

if Question.ask("Delete the table? (y/n) ", method(:is_yesno))
  movies.delete_table
  puts("Deleted #{table_name}.")
else
  puts("Don't forget to delete the table when you're done or you might incur \"\
    charges on your account.")
end

puts("\nThanks for watching!")
puts("-" * 88)
rescue Aws::Errors::ServiceError
  puts("Something went wrong with the demo.")
rescue Errno::ENOENT
  true
end

run_scenario(
  Movies.new(Aws::DynamoDB::Resource.new),
  "doc-example-table-movies", "moviedata.json",
  ) if $PROGRAM_NAME == __FILE__

```

This scenario uses the following helper class to ask questions at a command prompt.

```

# Asks a single question and validates it against a list of validators.
# When an answer fails validation, the complaint is printed and the question
# is asked again.
#
# @param question [String] The question to ask.
# @param validators [Array] The list of validators that the answer must pass.
# @return The answer, converted to its final form by the validators.
class Question
  def self.ask(question, *validators)
    answer = nil
    while answer.nil?
      puts(question)
      answer = gets.chomp
      validators.unshift(method(:non_empty)) unless validators[0] == method(:non_empty)
      validators.each do |validator|
        answer, complaint = validator.call(answer)
        if answer.nil?
          puts(complaint)
          break
        end
      end
    end
    answer
  end
end

# Validates that the answer is not empty.
# @return [Array] The non-empty answer, or nil.
def non_empty(answer)
  answer = nil unless answer != ""
  [answer, "I need an answer. Please?"]
end

# Validates a yes/no answer.
# @return [Array] True when the answer is 'y'; otherwise, False.
def is_yesno(answer)
  [answer.downcase == "y", ""]

```

```
end

# Validates that the answer can be converted to an int.
# @return [Array] The int answer; otherwise, nil.
def is_int(answer)
  int_answer = answer.to_i
  if int_answer == 0
    int_answer = nil
  end
  [int_answer, "#{answer} must be a valid integer."]
end

# Validates that the answer can be converted to a float.
# :return [Array] The float answer; otherwise, None.
def is_float(answer)
  float_answer = answer.to_f
  if float_answer == 0.0
    float_answer = nil
  end
  [float_answer, "#{answer} must be a valid float."]
end

# Validates that the answer is within a range. The answer must be of a type that can
# be compared to the lower and upper bounds.
# @return [Proc] A Proc that can be called to determine whether the answer is within
#               the expected range.
def in_range(lower, upper)
  Proc.new { |answer|
    answer.between?(lower, upper) ? range_answer = answer : range_answer = nil
    [range_answer, "#{answer} must be between #{lower} and #{upper}."]
  }
end
```

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## EventBridge examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Amazon EventBridge.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Scenarios \(p. 4325\)](#)

## Scenarios

### Create and trigger a rule

The following code example shows how to create and trigger a rule in Amazon EventBridge.

#### SDK for Ruby

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Call the functions in the correct order.

```
require "aws-sdk-sns"
require "aws-sdk-iam"
require "aws-sdk-cloudwatchevents"
require "aws-sdk-ec2"
require "aws-sdk-cloudwatch"
require "aws-sdk-cloudwatchlogs"
require "securerandom"
```

Checks whether the specified Amazon Simple Notification Service (Amazon SNS) topic exists among those provided to this function.

```
# Checks whether the specified Amazon SNS
# topic exists among those provided to this function.
# This is a helper function that is called by the topic_exists? function.
#
# @param topics [Array] An array of Aws::SNS::Types::Topic objects.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic')
#   )
#     puts 'Topic found.'
#   end

def topic_found?(topics, topic_arn)
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  return false
end
```

Checks whether the specified topic exists among those available to the caller in Amazon SNS.

```
# Checks whether the specified topic exists among those available to the
# caller in Amazon SNS.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   exit 1 unless topic_exists?(Aws::SNS::Client.new(region: 'us-east-1'),
```

```
#      'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#  )
def topic_exists?(sns_client, topic_arn)
  puts "Searching for topic with ARN '#{topic_arn}'..."
  response = sns_client.list_topics
  if response.topics.count.positive?
    if topic_found?(response.topics, topic_arn)
      puts "Topic found."
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.topics.count.positive?
        if topic_found?(response.topics, topic_arn)
          puts "Topic found."
          return true
        end
      end
    end
  end
  puts "Topic not found."
  return false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  return false
end
```

Create a topic in Amazon SNS and then subscribe an email address to receive notifications to that topic.

```
# Creates a topic in Amazon SNS
# and then subscribes an email address to receive notifications to that topic.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_name [String] The name of the topic to create.
# @param email_address [String] The email address of the recipient to notify.
# @return [String] The ARN of the topic that was created.
# @example
#   puts create_topic(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
  puts "Topic created with ARN '#{topic_response.topic_arn}'."
  subscription_response = sns_client.subscribe(
    topic_arn: topic_response.topic_arn,
    protocol: "email",
    endpoint: email_address,
    return_subscription_arn: true
  )
  puts "Subscription created with ARN " \
    "'#{subscription_response.subscription_arn}'. Have the owner of the " \
    "email address '#{email_address}' check their inbox in a few minutes " \
    "and confirm the subscription to start receiving notification emails."
  return topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  return "Error"
end
```

Check whether the specified AWS Identity and Access Management (IAM) role exists among those provided to this function.

```
# Checks whether the specified AWS Identity and Access Management (IAM)
# role exists among those provided to this function.
# This is a helper function that is called by the role_exists? function.
#
# @param roles [Array] An array of Aws::IAM::Role objects.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-east-1')
#   response = iam_client.list_roles
#   if role_found?(response.roles,
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
#     puts 'Role found.'
#   end
def role_found?(roles, role_arn)
  roles.each do |role|
    return true if role.arn == role_arn
  end
  return false
end
```

Check whether the specified role exists among those available to the caller in IAM.

```
# Checks whether the specified role exists among those available to the
# caller in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   exit 1 unless role_exists?(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
def role_exists?(iam_client, role_arn)
  puts "Searching for role with ARN '#{role_arn}'..."
  response = iam_client.list_roles
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts "Role found."
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.roles.count.positive?
        if role_found?(response.roles, role_arn)
          puts "Role found."
          return true
        end
      end
    end
  end
  puts "Role not found."
  return false
rescue StandardError => e
  puts "Role not found: #{e.message}"
  return false
end
```

Create a role in IAM.

```
# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon EventBridge to allow
# that rule to operate within the caller's account.
# This role is designed to be used specifically by this code example.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to create.
# @return [String] The ARN of the role that was created.
# @example
#   puts create_role(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
      'Version': "2012-10-17",
      'Statement': [
        {
          'Sid': "",
          'Effect': "Allow",
          'Principal': {
            'Service': "events.amazonaws.com"
          },
          'Action': "sts:AssumeRole"
        }
      ]
    }.to_json,
    path: "/",
    role_name: role_name
  )
  puts "Role created with ARN '#{response.role.arn}'."
  puts "Adding access policy to role..."
  iam_client.put_role_policy(
    policy_document: {
      'Version': "2012-10-17",
      'Statement': [
        {
          'Sid': "CloudWatchEventsFullAccess",
          'Effect': "Allow",
          'Resource': "*",
          'Action': "events:)"
        },
        {
          'Sid': "IAMPassRoleForCloudWatchEvents",
          'Effect': "Allow",
          'Resource': "arn:aws:iam::*:role/AWS_Events_Invoke_Targets",
          'Action': "iam:PassRole"
        }
      ]
    }.to_json,
    policy_name: "CloudWatchEventsPolicy",
    role_name: role_name
  )
  puts "Access policy added to role."
  return response.role.arn
rescue StandardError => e
  puts "Error creating role or adding policy to it: #{e.message}"
  puts "If the role was created, you must add the access policy \" \
    \"to the role yourself, or delete the role yourself and try again."
```

```
    return "Error"
end
```

Checks whether the specified EventBridge rule exists among those provided to this function.

```
# Checks whether the specified Amazon EventBridge rule exists among
# those provided to this function.
# This is a helper function that is called by the rule_exists? function.
#
# @param rules [Array] An array of Aws::CloudWatchEvents::Types::Rule objects.
# @param rule_arn [String] The name of the rule to find.
# @return [Boolean] true if the name of the rule was found; otherwise, false.
# @example
#   cloudwatchevents_client = Aws::CloudWatch::Client.new(region: 'us-east-1')
#   response = cloudwatchevents_client.list_rules
#   if rule_found?(response.rules, 'aws-doc-sdk-examples-ec2-state-change')
#     puts 'Rule found.'
#   end
def rule_found?(rules, rule_name)
  rules.each do |rule|
    return true if rule.name == rule_name
  end
  return false
end
```

Checks whether the specified rule exists among those available to the caller in EventBridge.

```
# Checks whether the specified rule exists among those available to the
# caller in Amazon EventBridge.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1')
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
  puts "Searching for rule with name '#{rule_name}'..."
  response = cloudwatchevents_client.list_rules
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts "Rule found."
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.rules.count.positive?
        if rule_found?(response.rules, rule_name)
          puts "Rule found."
          return true
        end
      end
    end
  end
  puts "Rule not found."
  return false
rescue StandardError => e
  puts "Rule not found: #{e.message}"
```

```
    return false
end
```

Create a rule in EventBridge.

```
# Creates a rule in Amazon EventBridge.
# This rule is triggered whenever an available instance in
# Amazon EC2 changes to the specified state.
# This rule is designed to be used specifically by this code example.
#
# Prerequisites:
#
# - A role in AWS Identity and Access Management (IAM) that is designed
#   to be used specifically by this code example.
# - A topic in Amazon SNS.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to create.
# @param rule_description [String] Some description for this rule.
# @param instance_state [String] The state that available instances in
#   Amazon EC2 must change to, to
#   trigger this rule.
# @param role_arn [String] The Amazon Resource Name (ARN) of the IAM role.
# @param target_id [String] Some identifying string for the rule's target.
# @param topic_arn [String] The ARN of the Amazon SNS topic.
# @return [Boolean] true if the rule was created; otherwise, false.
# @example
#   exit 1 unless rule_created?
#   Aws::CloudWatch::Client.new(region: 'us-east-1'),
#   'aws-doc-sdk-examples-ec2-state-change',
#   'Triggers when any available EC2 instance starts.',
#   'running',
#   'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change',
#   'sns-topic',
#   'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def rule_created?(  
  cloudwatchevents_client,  
  rule_name,  
  rule_description,  
  instance_state,  
  role_arn,  
  target_id,  
  topic_arn  
)  
  puts "Creating rule with name '#{rule_name}'..."  
  put_rule_response = cloudwatchevents_client.put_rule(  
    name: rule_name,  
    description: rule_description,  
    event_pattern: {  
      'source': [  
        "aws.ec2"  
      ],  
      'detail-type': [  
        "EC2 Instance State-change Notification"  
      ],  
      'detail': {  
        'state': [  
          instance_state  
        ]  
      }  
    }.to_json,  
    state: "ENABLED",
```

```

        role_arn: role_arn
    )
puts "Rule created with ARN '#{put_rule_response.rule_arn}'."

put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
    targets: [
        {
            id: target_id,
            arn: topic_arn
        }
    ]
)
if put_targets_response.key?(:failed_entry_count) &&
    put_targets_response.failed_entry_count > 0
    puts "Error(s) adding target to rule:"
    put_targets_response.failed_entries.each do |failure|
        puts failure.error_message
    end
    return false
else
    return true
end
rescue StandardError => e
    puts "Error creating rule or adding target to rule: #{e.message}"
    puts "If the rule was created, you must add the target " \
        "to the rule yourself, or delete the rule yourself and try again."
    return false
end

```

Check to see whether the specified log group exists among those available to the caller in Amazon CloudWatch Logs.

```

# Checks to see whether the specified log group exists among those available
# to the caller in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to find.
# @return [Boolean] true if the log group name was found; otherwise, false.
# @example
#   exit 1 unless log_group_exists?(

#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_exists?(cloudwatchlogs_client, log_group_name)
    puts "Searching for log group with name '#{log_group_name}'..."
    response = cloudwatchlogs_client.describe_log_groups(
        log_group_name_prefix: log_group_name
    )
    if response.log_groups.count.positive?
        response.log_groups.each do |log_group|
            if log_group.log_group_name == log_group_name
                puts "Log group found."
                return true
            end
        end
    end
    puts "Log group not found."
    return false
rescue StandardError => e
    puts "Log group not found: #{e.message}"
    return false
end

```

```
end
```

Create a log group in CloudWatch Logs.

```
# Creates a log group in Amazon CloudWatch Logs.  
#  
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized  
#   Amazon CloudWatch Logs client.  
# @param log_group_name [String] The name of the log group to create.  
# @return [Boolean] true if the log group name was created; otherwise, false.  
# @example  
#   exit 1 unless log_group_created?  
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),  
#     'aws-doc-sdk-examples-cloudwatch-log'  
#   )  
def log_group_created?(cloudwatchlogs_client, log_group_name)  
  puts "Attempting to create log group with the name '#{log_group_name}'..."  
  cloudwatchlogs_client.create_log_group(log_group_name: log_group_name)  
  puts "Log group created."  
  return true  
rescue StandardError => e  
  puts "Error creating log group: #{e.message}"  
  return false  
end
```

Write an event to a log stream in CloudWatch Logs.

```
# Writes an event to a log stream in Amazon CloudWatch Logs.  
#  
# Prerequisites:  
#  
# - A log group in Amazon CloudWatch Logs.  
# - A log stream within the log group.  
#  
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized  
#   Amazon CloudWatch Logs client.  
# @param log_group_name [String] The name of the log group.  
# @param log_stream_name [String] The name of the log stream within  
#   the log group.  
# @param message [String] The message to write to the log stream.  
# @param sequence_token [String] If available, the sequence token from the  
#   message that was written immediately before this message. This sequence  
#   token is returned by Amazon CloudWatch Logs whenever you programmatically  
#   write a message to the log stream.  
# @return [String] The sequence token that is returned by  
#   Amazon CloudWatch Logs after successfully writing the message to the  
#   log stream.  
# @example  
#   puts log_event(  
#     Aws::EC2::Client.new(region: 'us-east-1'),  
#     'aws-doc-sdk-examples-cloudwatch-log'  
#     '2020/11/19/53f985be-199f-408e-9a45-fc242df41fEX',  
#     "Instance 'i-033c48ef067af3dEX' restarted.",  
#     '495426724868310740095796045676567882148068632824696073EX'  
#   )  
def log_event(  
  cloudwatchlogs_client,  
  log_group_name,  
  log_stream_name,  
  message,  
  sequence_token
```

```

)
  puts "Attempting to log '#{message}' to log stream '#{log_stream_name}'..."
  event = {
    log_group_name: log_group_name,
    log_stream_name: log_stream_name,
    log_events: [
      {
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,
        message: message
      }
    ]
  }
unless sequence_token.empty?
  event[:sequence_token] = sequence_token
end

response = cloudwatchlogs_client.put_log_events(event)
puts "Message logged."
return response.next_sequence_token
rescue StandardError => e
  puts "Message not logged: #{e.message}"
end

```

Restart an Amazon Elastic Compute Cloud (Amazon EC2) instance and adds information about the related activity to a log stream in CloudWatch Logs.

```

# Restarts an Amazon EC2 instance
# and adds information about the related activity to a log stream
# in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - The Amazon EC2 instance to restart.
# - The log group in Amazon CloudWatch Logs to add related activity
#   information to.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client]
#   An initialized Amazon CloudWatch Logs client.
# @param instance_id [String] The ID of the instance.
# @param log_group_name [String] The name of the log group.
# @return [Boolean] true if the instance was restarted and the information
#   was written to the log stream; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX',
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def instance_restarted?(  

  ec2_client,  

  cloudwatchlogs_client,  

  instance_id,  

  log_group_name
)
  log_stream_name = "#{Time.now.year}/#{Time.now.month}/#{Time.now.day}/" \
    "#{SecureRandom.uuid}"
  cloudwatchlogs_client.create_log_stream(
    log_group_name: log_group_name,
    log_stream_name: log_stream_name
)

```

```

sequence_token = ""

puts "Attempting to stop the instance with the ID '#{instance_id}'. " \
    "This might take a few minutes..."
ec2_client.stop_instances(instance_ids: [instance_id])
ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
puts "Instance stopped."
sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' stopped.",
    sequence_token
)

puts "Attempting to restart the instance. This might take a few minutes..."
ec2_client.start_instances(instance_ids: [instance_id])
ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
puts "Instance restarted."
sequence_token = log_event(
    cloudwatchlogs_client,
    log_group_name,
    log_stream_name,
    "Instance '#{instance_id}' restarted.",
    sequence_token
)

return true
rescue StandardError => e
    puts "Error creating log stream or stopping or restarting the instance: " \
        "#{e.message}"
    log_event(
        cloudwatchlogs_client,
        log_group_name,
        log_stream_name,
        "Error stopping or starting instance '#{instance_id}': #{e.message}",
        sequence_token
    )
    return false
end

```

Display information about activity for a rule in EventBridge.

```

# Displays information about activity for a rule in Amazon EventBridge.
#
# Prerequisites:
#
# - A rule in Amazon EventBridge.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example
#   display_rule_activity(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change',
#     Time.now - 600, # Start checking from 10 minutes ago.
#     Time.now, # Check up until now.
#     60 # Check every minute during those 10 minutes.

```

```

#     )
def display_rule_activity(
    cloudwatch_client,
    rule_name,
    start_time,
    end_time,
    period
)
    puts "Attempting to display rule activity..."
    response = cloudwatch_client.get_metric_statistics(
        namespace: "AWS/Events",
        metric_name: "Invocations",
        dimensions: [
            {
                name: "RuleName",
                value: rule_name
            }
        ],
        start_time: start_time,
        end_time: end_time,
        period: period,
        statistics: ["Sum"],
        unit: "Count"
    )

    if response.key?(:datapoints) && response.datapoints.count.positive?
        puts "The event rule '#{rule_name}' was triggered:"
        response.datapoints.each do |datapoint|
            puts " #{datapoint.sum} time(s) at #{datapoint.timestamp}"
        end
    else
        puts "The event rule '#{rule_name}' was not triggered during the " \
            "specified time period."
    end
rescue StandardError => e
    puts "Error getting information about event rule activity: #{e.message}"
end

```

Display log information for all of the log streams in a CloudWatch Logs log group.

```

# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example
#   display_log_data(
#     Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def display_log_data(cloudwatchlogs_client, log_group_name)
    puts "Attempting to display log stream data for the log group " \
        "named '#{log_group_name}'..."
    describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
        log_group_name: log_group_name,
        order_by: "LastEventTime",
        descending: true
    )
    if describe_log_streams_response.key?(:log_streams) &&

```

```
    describe_log_streams_response.log_streams.count.positive?
    describe_log_streams_response.log_streams.each do |log_stream|
      get_log_events_response = cloudwatchlogs_client.get_log_events(
        log_group_name: log_group_name,
        log_stream_name: log_stream.log_stream_name
      )
      puts "\nLog messages for '#{log_stream.log_stream_name}':"
      puts "-" * (log_stream.log_stream_name.length + 20)
      if get_log_events_response.key?(:events) &&
          get_log_events_response.events.count.positive?
        get_log_events_response.events.each do |event|
          puts event.message
        end
      else
        puts "No log messages for this log stream."
      end
    end
  rescue StandardError => e
    puts "Error getting information about the log streams or their messages: \" \
      \"#{e.message}\""
end
```

Display a reminder to the caller to manually clean up any associated AWS resources that they no longer need.

```
# Displays a reminder to the caller to manually clean up any associated
# AWS resources that they no longer need.
#
# @param topic_name [String] The name of the Amazon SNS topic.
# @param role_name [String] The name of the IAM role.
# @param rule_name [String] The name of the Amazon EventBridge rule.
# @param log_group_name [String] The name of the Amazon CloudWatch Logs log group.
# @param instance_id [String] The ID of the Amazon EC2 instance.
# @example
#   manual_cleanup_notice(
#     'aws-doc-sdk-examples-topic',
#     'aws-doc-sdk-examples-cloudwatch-events-rule-role',
#     'aws-doc-sdk-examples-ec2-state-change',
#     'aws-doc-sdk-examples-cloudwatch-log',
#     'i-033c48ef067af3dEX'
#   )
def manual_cleanup_notice(
  topic_name, role_name, rule_name, log_group_name, instance_id
)
  puts "-" * 10
  puts "Some of the following AWS resources might still exist in your account."
  puts "If you no longer want to use this code example, then to clean up"
  puts "your AWS account and avoid unexpected costs, you might want to"
  puts "manually delete any of the following resources if they exist:"
  puts "- The Amazon SNS topic named '#{topic_name}'."
  puts "- The IAM role named '#{role_name}'."
  puts "- The Amazon EventBridge rule named '#{rule_name}'."
  puts "- The Amazon CloudWatch Logs log group named '#{log_group_name}'."
  puts "- The Amazon EC2 instance with the ID '#{instance_id}'."
end

# Full example call:
def run_me
  # Properties for the Amazon SNS topic.
  topic_name = "aws-doc-sdk-examples-topic"
  email_address = "mary@example.com"
```

```
# Properties for the IAM role.
role_name = "aws-doc-sdk-examples-cloudwatch-events-rule-role"
# Properties for the Amazon EventBridge rule.
rule_name = "aws-doc-sdk-examples-ec2-state-change"
rule_description = "Triggers when any available EC2 instance starts."
instance_state = "running"
target_id = "sns-topic"
# Properties for the Amazon EC2 instance.
instance_id = "i-033c48ef067af3dEX"
# Properties for displaying the event rule's activity.
start_time = Time.now - 600 # Go back over the past 10 minutes
                           # (10 minutes * 60 seconds = 600 seconds).
end_time = Time.now
period = 60 # Look back every 60 seconds over the past 10 minutes.
# Properties for the Amazon CloudWatch Logs log group.
log_group_name = "aws-doc-sdk-examples-cloudwatch-log"
# AWS service clients for this code example.
region = "us-east-1"
sts_client = Aws::STS::Client.new(region: region)
sns_client = Aws::SNS::Client.new(region: region)
iam_client = Aws::IAM::Client.new(region: region)
cloudwatchevents_client = Aws::CloudWatchEvents::Client.new(region: region)
ec2_client = Aws::EC2::Client.new(region: region)
cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
cloudwatchlogs_client = Aws::CloudWatchLogs::Client.new(region: region)

# Get the caller's account ID for use in forming
# Amazon Resource Names (ARNs) that this code relies on later.
account_id = sts_client.get_caller_identity.account

# If the Amazon SNS topic doesn't exist, create it.
topic_arn = "arn:aws:sns:{region}:{account_id}:{topic_name}"
unless topic_exists?(sns_client, topic_arn)
    topic_arn = create_topic(sns_client, topic_name, email_address)
    if topic_arn == "Error"
        puts "Could not create the Amazon SNS topic correctly. Program stopped."
        manual_cleanup_notice(
            topic_name, role_name, rule_name, log_group_name, instance_id
        )
        exit 1
    end
end

# If the IAM role doesn't exist, create it.
role_arn = "arn:aws:iam:{account_id}:role/{role_name}"
unless role_exists?(iam_client, role_arn)
    role_arn = create_role(iam_client, role_name)
    if role_arn == "Error"
        puts "Could not create the IAM role correctly. Program stopped."
        manual_cleanup_notice(
            topic_name, role_name, rule_name, log_group_name, instance_id
        )
    end
end

# If the Amazon EventBridge rule doesn't exist, create it.
unless rule_exists?(cloudwatchevents_client, rule_name)
    unless rule_created?(
        cloudwatchevents_client,
        rule_name,
        rule_description,
        instance_state,
        role_arn,
        target_id,
        topic_arn
    )

```

```
    puts "Could not create the Amazon EventBridge rule correctly. " \
        "Program stopped."
    manual_cleanup_notice(
        topic_name, role_name, rule_name, log_group_name, instance_id
    )
end
end

# If the Amazon CloudWatch Logs log group doesn't exist, create it.
unless log_group_exists?(cloudwatchlogs_client, log_group_name)
    unless log_group_created?(cloudwatchlogs_client, log_group_name)
        puts "Could not create the Amazon CloudWatch Logs log group " \
            "correctly. Program stopped."
        manual_cleanup_notice(
            topic_name, role_name, rule_name, log_group_name, instance_id
        )
    end
end

# Restart the Amazon EC2 instance, which triggers the rule.
unless instance_restarted?(
    ec2_client,
    cloudwatchlogs_client,
    instance_id,
    log_group_name
)
    puts "Could not restart the instance to trigger the rule. " \
        "Continuing anyway to show information about the rule and logs..."
end

# Display how many times the rule was triggered over the past 10 minutes.
display_rule_activity(
    cloudwatch_client,
    rule_name,
    start_time,
    end_time,
    period
)

# Display related log data in Amazon CloudWatch Logs.
display_log_data(cloudwatchlogs_client, log_group_name)

# Reminder the caller to clean up any AWS resources that are used
# by this code example and are no longer needed.
manual_cleanup_notice(
    topic_name, role_name, rule_name, log_group_name, instance_id
)
end

run_me if $PROGRAM_NAME == __FILE__
```

## IAM examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with AWS Identity and Access Management.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4339\)](#)
- [Scenarios \(p. 4352\)](#)

## Actions

### Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

#### SDK for Ruby

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates a policy that grants permission to list S3 buckets in the account, and
# then attaches the policy to a role.
#
# @param policy_name [String] The name to give the policy.
# @param role [Aws::IAM::Role] The role that the policy is attached to.
# @return [Aws::IAM::Policy] The newly created policy.
def create_and_attach_role_policy(policy_name, role)
    policy = @iam_resource.create_policy(
        policy_name: policy_name,
        policy_document: {
            Version: "2012-10-17",
            Statement: [
                {
                    Effect: "Allow",
                    Action: "s3>ListAllMyBuckets",
                    Resource: "arn:aws:s3:::*"
                }
            ]
        }.to_json
    role.attach_policy(policy_arn: policy.arn)
    puts("Created policy #{policy.policy_name} and attached it to role #{role.name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't create a policy and attach it to role #{role.name}. Here's why: ")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    policy
end
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Ruby API Reference*.

### Create a policy

The following code example shows how to create an IAM policy.

#### SDK for Ruby

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates a policy that grants permission to list S3 buckets in the account, and
# then attaches the policy to a role.
#
```

```
# @param policy_name [String] The name to give the policy.
# @param role [Aws::IAM::Role] The role that the policy is attached to.
# @return [Aws::IAM::Policy] The newly created policy.
def create_and_attach_role_policy(policy_name, role)
    policy = @iam_resource.create_policy(
        policy_name: policy_name,
        policy_document: {
            Version: "2012-10-17",
            Statement: [
                {
                    Effect: "Allow",
                    Action: "s3>ListAllMyBuckets",
                    Resource: "arn:aws:s3:::*"
                }
            ].to_json
    )
    role.attach_policy(policy_arn: policy.arn)
    puts("Created policy #{policy.policy_name} and attached it to role #{role.name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't create a policy and attach it to role #{role.name}. Here's why: ")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    policy
end
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Ruby API Reference*.

## Create a role

The following code example shows how to create an IAM role.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates a role that can be assumed by a user.
#
# @param role_name [String] The name to give the role.
# @param user [Aws::IAM::User] The user who is granted permission to assume the role.
# @return [Aws::IAM::Role] The newly created role.
def create_role(role_name, user)
    role = @iam_resource.create_role(
        role_name: role_name,
        assume_role_policy_document: {
            Version: "2012-10-17",
            Statement: [
                {
                    Effect: "Allow",
                    Principal: {'AWS': user.arn},
                    Action: "sts:AssumeRole"
                }
            ].to_json
    )
    puts("Created role #{role.name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't create a role for the demo. Here's why: ")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    role
end
```

- For API details, see [CreateRole](#) in *AWS SDK for Ruby API Reference*.

## Create a service-linked role

The following code example shows how to create an IAM service-linked role.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates a service-linked role.
#
# @param service_name [String] The name of the service that owns the role.
# @param description [String] A description to give the role.
# @return [Aws::IAM::Role] The newly created role.
def create_service_linked_role(service_name, description)
    response = @iam_resource.client.create_service_linked_role(
        aws_service_name: service_name, description: description)
    role = @iam_resource.role(response.role.role_name)
    puts("Created service-linked role #{role.name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't create service-linked role for #{service_name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    role
end
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for Ruby API Reference*.

## Create a user

The following code example shows how to create an IAM user.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates a user.
#
# @param user_name [String] The name to give the user.
# @return [Aws::IAM::User] The newly created user.
def create_user(user_name)
    user = @iam_resource.create_user(user_name: user_name)
    puts("Created demo user named #{user.name}.")
rescue Aws::Errors::ServiceError => e
    puts("Tried and failed to create demo user.")
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a user!")
    raise
else
    user
end
```

- For API details, see [CreateUser](#) in *AWS SDK for Ruby API Reference*.

## Create an access key

The following code example shows how to create an IAM access key.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates an access key for a user.
#
# @param user [Aws::IAM::User] The user that owns the key.
# @return [Aws::IAM::AccessKeyPair] The newly created access key.
def create_access_key_pair(user)
    user_key = user.create_access_key_pair
    puts("Created access key pair for user.")
    rescue Aws::Errors::ServiceError => e
        puts("Couldn't create access keys for user #{user.name}.")
        puts("\t#{e.code}: #{e.message}")
        raise
    else
        user_key
    end
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for Ruby API Reference*.

## Create an inline policy for a user

The following code example shows how to create an inline IAM policy for a user.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates an inline policy for a user that lets the user assume a role.
#
# @param policy_name [String] The name to give the policy.
# @param user [Aws::IAM::User] The user that owns the policy.
# @param role [Aws::IAM::Role] The role that can be assumed.
# @return [Aws::IAM::UserPolicy] The newly created policy.
def create_user_policy(policy_name, user, role)
    policy = user.create_policy(
        policy_name: policy_name,
        policy_document: {
            Version: "2012-10-17",
            Statement: [
                {
                    Effect: "Allow",
                    Action: "sts:AssumeRole",
                    Resource: role.arn
                }
            ].to_json
        }
    )
    puts("Created an inline policy for #{user.name} that lets the user assume role #{role.name}.")
```

```
rescue Aws::Errors::ServiceError => e
    puts("Couldn't create an inline policy for user #{user.name}. Here's why: ")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    policy
end
```

- For API details, see [PutUserPolicy](#) in *AWS SDK for Ruby API Reference*.

## Delete a policy

The following code example shows how to delete an IAM policy.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a role. If the role has policies attached, they are detached and
# deleted before the role is deleted.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_role(role)
    role.attached_policies.each do |policy|
        name = policy.policy_name
        policy.detach_role(role_name: role.name)
        policy.delete
        puts("Deleted policy #{name}.")
    end
    name = role.name
    role.delete
    puts("Deleted role #{name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't detach policies and delete role #{role.name}. Here's why: ")
    puts("\t#{e.code}: #{e.message}")
    raise
end
```

- For API details, see [DeletePolicy](#) in *AWS SDK for Ruby API Reference*.

## Delete a role

The following code example shows how to delete an IAM role.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a role. If the role has policies attached, they are detached and
# deleted before the role is deleted.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_role(role)
```

```
role.attached_policies.each do |policy|
  name = policy.policy_name
  policy.detach_role(role_name: role.name)
  policy.delete
  puts("Deleted policy #{name}.")
end
name = role.name
role.delete
puts("Deleted role #{name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't detach policies and delete role #{role.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- For API details, see [DeleteRole](#) in *AWS SDK for Ruby API Reference*.

## Delete a service-linked role

The following code example shows how to delete an IAM service-linked role.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a service-linked role from the account.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_service_linked_role(role)
  response = @iam_resource.client.delete_service_linked_role(role_name: role.name)
  task_id = response.deletion_task_id
  while true
    response = @iam_resource.client.get_service_linked_role_deletion_status(
      deletion_task_id: task_id)
    status = response.status
    puts("Deletion of #{role.name} #{status}.")
    if %w[SUCCEEDED FAILED].include?(status)
      break
    else
      sleep(3)
    end
  end
rescue Aws::Errors::ServiceError => e
  # If AWS has not yet fully propagated the role, it deletes the role but
  # returns NoSuchEntity.
  if e.code != "NoSuchEntity"
    puts("Couldn't delete #{role.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- For API details, see [DeleteServiceLinkedRole](#) in *AWS SDK for Ruby API Reference*.

## Delete a user

The following code example shows how to delete an IAM user.

## SDK for Ruby

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a user. If the user has inline policies or access keys, they are deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
    user.policies.each do |policy|
        name = policy.name
        policy.delete
        puts("Deleted user policy #{name}.")
    end
    user.access_keys.each do |key|
        key.delete
        puts("Deleted access key for user #{user.name}.")
    end
    name = user.name
    user.delete
    puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
end
```

- For API details, see [DeleteUser](#) in *AWS SDK for Ruby API Reference*.

## Delete an access key

The following code example shows how to delete an IAM access key.

## SDK for Ruby

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a user. If the user has inline policies or access keys, they are deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
    user.policies.each do |policy|
        name = policy.name
        policy.delete
        puts("Deleted user policy #{name}.")
    end
    user.access_keys.each do |key|
        key.delete
        puts("Deleted access key for user #{user.name}.")
    end
    name = user.name
    user.delete
    puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
end
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for Ruby API Reference*.

## Delete an inline policy from a user

The following code example shows how to delete an inline IAM policy from a user.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a user. If the user has inline policies or access keys, they are deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
    user.policies.each do |policy|
        name = policy.name
        policy.delete
        puts("Deleted user policy #{name}.")
    end
    user.access_keys.each do |key|
        key.delete
        puts("Deleted access key for user #{user.name}.")
    end
    name = user.name
    user.delete
    puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
end
```

- For API details, see [DeleteUserPolicy](#) in *AWS SDK for Ruby API Reference*.

## Detach a policy from a role

The following code example shows how to detach an IAM policy from a role.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a role. If the role has policies attached, they are detached and
# deleted before the role is deleted.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_role(role)
    role.attached_policies.each do |policy|
        name = policy.policy_name
        policy.detach_role(role_name: role.name)
        policy.delete
        puts("Deleted policy #{name}.")
    end
```

```
name = role.name
role.delete
puts("Deleted role #{name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't detach policies and delete role #{role.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- For API details, see [DetachRolePolicy](#) in *AWS SDK for Ruby API Reference*.

## Get a policy

The following code example shows how to get an IAM policy.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Gets data about a policy.
#
# @param policy_arn [String] The ARN of the policy to look up.
# @return [Aws::IAM::Policy] The retrieved policy.
def get_policy(policy_arn)
  policy = @iam_resource.policy(policy_arn)
  puts("Got policy '#{policy.policy_name}'. Its ID is: #{policy.policy_id}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get policy '#{policy_arn}'. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  policy
end
```

- For API details, see [GetPolicy](#) in *AWS SDK for Ruby API Reference*.

## Get a role

The following code example shows how to get an IAM role.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Gets data about a role.
#
# @param name [String] The name of the role to look up.
# @return [Aws::IAM::Role] The retrieved role.
def get_role(name)
  role = @iam_resource.role(name)
  puts("Got data for role '#{role.name}'. Its ARN is '#{role.arn}' .")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get data for role '#{name}' Here's why:")
  puts("\t#{e.code}: #{e.message}")
```

```
    raise
else
  role
end
```

- For API details, see [GetRole](#) in *AWS SDK for Ruby API Reference*.

## Get the account password policy

The following code example shows how to get the IAM account password policy.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Prints the password policy for the account.
def print_account_password_policy
  policy = @iam_resource.account_password_policy
  policy.load
  puts("The account password policy is:")
  puts(policy.data.to_h)
rescue Aws::Errors::ServiceError => e
  if e.code == "NoSuchEntity"
    puts("The account does not have a password policy.")
  else
    puts("Couldn't print the account password policy. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for Ruby API Reference*.

## List SAML providers

The following code example shows how to list SAML providers for IAM.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Lists up to a specified number of SAML providers for the account.
#
# @param count [Integer] The maximum number of providers to list.
def list_saml_providers(count)
  @iam_resource.saml_providers.limit(count).each do |provider|
    puts("\t#{provider.arn}")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't list SAML providers. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for Ruby API Reference*.

## List a user's access keys

The following code example shows how to list a user's IAM access keys.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a user. If the user has inline policies or access keys, they are deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
    user.policies.each do |policy|
        name = policy.name
        policy.delete
        puts("Deleted user policy #{name}.")
    end
    user.access_keys.each do |key|
        key.delete
        puts("Deleted access key for user #{user.name}.")
    end
    name = user.name
    user.delete
    puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
end
```

- For API details, see [ListAccessKeys](#) in *AWS SDK for Ruby API Reference*.

## List groups

The following code example shows how to list IAM groups.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Lists up to a specified number of groups for the account.
#
# @param count [Integer] The maximum number of groups to list.
def list_groups(count)
    @iam_resource.groups.limit(count).each do |group|
        puts("\t#{group.name}")
    end
rescue Aws::Errors::ServiceError => e
    puts("Couldn't list groups for the account. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
end
```

- For API details, see [ListGroup](#) in *AWS SDK for Ruby API Reference*.

## List policies

The following code example shows how to list IAM policies.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Lists up to a specified number of policies in the account.  
#  
# @param count [Integer] The maximum number of policies to list.  
# @return [Array] The Amazon Resource Names (ARNs) of the policies listed.  
def list_policies(count)  
    policy_arns = []  
    @iam_resource.policies.limit(count).each_with_index do |policy, index|  
        puts("\t#{index + 1}: #{policy.policy_name}: #{policy.arn}")  
        policy_arns.append(policy.arn)  
    end  
    rescue Aws::Errors::ServiceError => e  
        puts("Couldn't list policies for the account. Here's why:")  
        puts("\t#{e.code}: #{e.message}")  
        raise  
    else  
        policy_arns  
    end
```

- For API details, see [ListPolicies](#) in *AWS SDK for Ruby API Reference*.

## List policies attached to a role

The following code example shows how to list policies attached to an IAM role.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a role. If the role has policies attached, they are detached and  
# deleted before the role is deleted.  
#  
# @param role [Aws::IAM::Role] The role to delete.  
def delete_role(role)  
    role.attached_policies.each do |policy|  
        name = policy.policy_name  
        policy.detach_role(role_name: role.name)  
        policy.delete  
        puts("Deleted policy #{name}.")  
    end  
    name = role.name  
    role.delete  
    puts("Deleted role #{name}.")  
rescue Aws::Errors::ServiceError => e  
    puts("Couldn't detach policies and delete role #{role.name}. Here's why:")  
    puts("\t#{e.code}: #{e.message}")
```

```
    raise
end
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for Ruby API Reference*.

## List roles

The following code example shows how to list IAM roles.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Lists up to a specified number of roles for the account.
#
# @param count [Integer] The maximum number of roles to list.
# @return [Array] The names of the listed roles.
def list_roles(count)
  role_names = []
  @iam_resource.roles.limit(count).each_with_index do |role, index|
    puts("\t#{index + 1}: #{role.name}")
    role_names.append(role.name)
  end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list roles for the account. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    role_names
  end
```

- For API details, see [ListRoles](#) in *AWS SDK for Ruby API Reference*.

## List users

The following code example shows how to list IAM users.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Lists up to a specified number of users in the account.
#
# @param count [Integer] The maximum number of users to list.
def list_users(count)
  @iam_resource.users.limit(count).each do |user|
    puts("\t#{user.name}")
  end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list users for the account. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- For API details, see [ListUsers](#) in *AWS SDK for Ruby API Reference*.

## Scenarios

### Create a user and assume a role

The following code example shows how to:

- Create a user who has no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list Amazon S3 buckets using temporary credentials.
- Delete the policy, role, and user.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-iam"
require "aws-sdk-s3"

# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_resource

  # @param iam_resource [Aws::IAM::Resource] An AWS IAM resource.
  def initialize(iam_resource)
    @iam_resource = iam_resource
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
    puts("Give AWS time to propagate resources...")
    sleep(duration)
  end

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_resource.create_user(user_name: user_name)
    puts("Created demo user named #{user.name}.")
    rescue Aws::Errors::ServiceError => e
      puts("Tried and failed to create demo user.")
      puts("\t#{e.code}: #{e.message}")
      puts("\nCan't continue the demo without a user!")
      raise
    else
      user
    end

    # Creates an access key for a user.
    #
```

```
# @param user [Aws::IAM::User] The user that owns the key.
# @return [Aws::IAM::AccessKeyPair] The newly created access key.
def create_access_key_pair(user)
    user_key = user.create_access_key_pair
    puts("Created access key pair for user.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't create access keys for user #{user.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    user_key
end

# Creates a role that can be assumed by a user.
#
# @param role_name [String] The name to give the role.
# @param user [Aws::IAM::User] The user who is granted permission to assume the role.
# @return [Aws::IAM::Role] The newly created role.
def create_role(role_name, user)
    role = @iam_resource.create_role(
        role_name: role_name,
        assume_role_policy_document: {
            Version: "2012-10-17",
            Statement: [
                {
                    Effect: "Allow",
                    Principal: {'AWS': user.arn},
                    Action: "sts:AssumeRole"
                }
            ].to_json)
    puts("Created role #{role.name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't create a role for the demo. Here's why: ")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    role
end

# Creates a policy that grants permission to list S3 buckets in the account, and
# then attaches the policy to a role.
#
# @param policy_name [String] The name to give the policy.
# @param role [Aws::IAM::Role] The role that the policy is attached to.
# @return [Aws::IAM::Policy] The newly created policy.
def create_and_attach_role_policy(policy_name, role)
    policy = @iam_resource.create_policy(
        policy_name: policy_name,
        policy_document: {
            Version: "2012-10-17",
            Statement: [
                {
                    Effect: "Allow",
                    Action: "s3>ListAllMyBuckets",
                    Resource: "arn:aws:s3:::/*"
                }
            ].to_json)
    role.attach_policy(policy_arn: policy.arn)
    puts("Created policy #{policy.policy_name} and attached it to role #{role.name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't create a policy and attach it to role #{role.name}. Here's why: ")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    policy
end

# Creates an inline policy for a user that lets the user assume a role.
```

```
#  
# @param policy_name [String] The name to give the policy.  
# @param user [Aws::IAM::User] The user that owns the policy.  
# @param role [Aws::IAM::Role] The role that can be assumed.  
# @return [Aws::IAM::UserPolicy] The newly created policy.  
def create_user_policy(policy_name, user, role)  
    policy = user.create_policy(  
        policy_name: policy_name,  
        policy_document: {  
            Version: "2012-10-17",  
            Statement: [{  
                Effect: "Allow",  
                Action: "sts:AssumeRole",  
                Resource: role.arn  
            }]  
        }.to_json)  
    puts("Created an inline policy for #{user.name} that lets the user assume role  
#{role.name}.")  
    rescue Aws::Errors::ServiceError => e  
        puts("Couldn't create an inline policy for user #{user.name}. Here's why: ")  
        puts("\t#{e.code}: #{e.message}")  
        raise  
    else  
        policy  
    end  
  
    # Creates an Amazon S3 resource with specified credentials. This is separated into a  
    # factory function so that it can be mocked for unit testing.  
    #  
    # @param credentials [Aws::Credentials] The credentials used by the Amazon S3  
    # resource.  
    def create_s3_resource(credentials)  
        Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))  
    end  
  
    # Lists the S3 buckets for the account, using the specified Amazon S3 resource.  
    # Because the resource uses credentials with limited access, it may not be able to  
    # list the S3 buckets.  
    #  
    # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.  
    def list_buckets(s3_resource)  
        count = 10  
        s3_resource.buckets.each do |bucket|  
            puts "\t#{bucket.name}"  
            count -= 1  
            break if count.zero?  
        end  
        rescue Aws::Errors::ServiceError => e  
            if e.code == "AccessDenied"  
                puts("Attempt to list buckets with no permissions: AccessDenied.")  
            else  
                puts("Couldn't list buckets for the account. Here's why: ")  
                puts("\t#{e.code}: #{e.message}")  
                raise  
            end  
    end  
  
    # Creates an AWS Security Token Service (AWS STS) client with specified credentials.  
    # This is separated into a factory function so that it can be mocked for unit  
    # testing.  
    #  
    # @param key_id [String] The ID of the access key used by the STS client.  
    # @param key_secret [String] The secret part of the access key used by the STS  
    # client.  
    def create_sts_client(key_id, key_secret)  
        Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
```

```
end

# Gets temporary credentials that can be used to assume a role.
#
# @param role_arn [String] The ARN of the role that is assumed when these credentials
#                               are used.
# @param sts_client [AWS::STS::Client] An AWS STS client.
# @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume the
role.
def assume_role(role_arn, sts_client)
  credentials = Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: "create-use-assume-role-scenario"
  )
  puts("Assumed role '#{role_arn}', got temporary credentials.")
  credentials
end

# Deletes a role. If the role has policies attached, they are detached and
# deleted before the role is deleted.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_role(role)
  role.attached_policies.each do |policy|
    name = policy.policy_name
    policy.detach_role(role_name: role.name)
    policy.delete
    puts("Deleted policy #{name}.")
  end
  name = role.name
  role.delete
  puts("Deleted role #{name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't detach policies and delete role #{role.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Deletes a user. If the user has inline policies or access keys, they are deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
  user.policies.each do |policy|
    name = policy.name
    policy.delete
    puts("Deleted user policy #{name}.")
  end
  user.access_keys.each do |key|
    key.delete
    puts("Deleted access key for user #{user.name}.")
  end
  name = user.name
  user.delete
  puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the IAM create a user and assume a role demo!")
end
```

```
puts("-" * 88)

user = scenario.create_user("doc-example-user-#{Random.uuid}")
user_key = scenario.create_access_key_pair(user)
scenario.wait(10)
role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
scenario.create_and_attach_role_policy("doc-example-role-policy-#{Random.uuid}", role)
scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user, role)
scenario.wait(10)
puts("Try to list buckets with credentials for a user who has no permissions.")
puts("Expect AccessDenied from this call.")
scenario.list_buckets()
scenario.create_s3_resource(Aws::Credentials.new(user_key.id, user_key.secret)))
puts("Now, assume the role that grants permission.")
temp_credentials = scenario.assume_role(
  role.arn, scenario.create_sts_client(user_key.id, user_key.secret))
puts("Here are your buckets:")
scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
puts("Deleting role '#{role.name}' and attached policies.")
scenario.delete_role(role)
puts("Deleting user '#{user.name}', policies, and keys.")
scenario.delete_user(user)

puts("Thanks for watching!")
puts("-" * 88)
rescue Aws::Errors::ServiceError => e
  puts("Something went wrong with the demo.")
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Resource.new)) if $PROGRAM_NAME
== __FILE__
```

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## Lambda examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with AWS Lambda.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Topics

- [Actions \(p. 4357\)](#)
- [Scenarios \(p. 4361\)](#)

## Actions

### Create a function

The following code example shows how to create a Lambda function.

#### SDK for Ruby

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function. This
  #                      must include the file name and the function name.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
  #                           code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
      role: role_arn.to_s,
      function_name:,
      handler: handler_name,
      runtime: "ruby2.7",
      code: {
        zip_file: deployment_package
      },
      environment: {
        variables: {
          "LOG_LEVEL" => "info"
        }
      }
    })
    @lambda_client.wait_until(:function_active_v2, { function_name: }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n#{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n#{e.message}")
  end
end
```

- For API details, see [CreateFunction](#) in *AWS SDK for Ruby API Reference*.

## Delete a function

The following code example shows how to delete a Lambda function.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deletes a Lambda function.
  # @param function_name: The name of the function to delete.
  def delete_function(function_name)
    print "Deleting function: #{function_name}..."
    @lambda_client.delete_function(
      function_name:
    )
    print "Done!".green
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
  end
```

- For API details, see [DeleteFunction](#) in *AWS SDK for Ruby API Reference*.

## Get a function

The following code example shows how to get a Lambda function.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Gets data about a Lambda function.
  #
  # @param function_name: The name of the function.
  # @return response: The function data, or nil if no such function exists.
  def get_function(function_name)
```

```
@lambda_client.get_function(
  {
    function_name:
  }
)
rescue Aws::Lambda::Errors::ResourceNotFoundException => e
  @logger.debug("Could not find function: #{function_name}:\n #{e.message}")
  nil
end
```

- For API details, see [GetFunction](#) in *AWS SDK for Ruby API Reference*.

## Invoke a function

The following code example shows how to invoke a Lambda function.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
  end
```

- For API details, see [Invoke](#) in *AWS SDK for Ruby API Reference*.

## List functions

The following code example shows how to list Lambda functions.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper
  attr_accessor :lambda_client
```

```
def initialize
  @lambda_client = Aws::Lambda::Client.new
  @logger = Logger.new($stdout)
  @logger.level = Logger::WARN
end

# Lists the Lambda functions for the current account.
def list_functions
  functions = []
  @lambda_client.list_functions.each do |response|
    response["functions"].each do |function|
      functions.append(function["function_name"])
    end
  end
  functions
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error executing #{function_name}:\n #{e.message}")
end
```

- For API details, see [ListFunctions](#) in *AWS SDK for Ruby API Reference*.

## Update function code

The following code example shows how to update Lambda function code.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the code for a Lambda function by submitting a .zip archive that contains
  # the code for the function.

  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                           .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name:,
      zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name: }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    rescue Aws::Lambda::Errors::ServiceException => e
      @logger.error("There was an error updating function code for: #{function_name}:\n#{e.message}")
      nil
    rescue Aws::Waiters::Errors::WaiterFailed => e
      @logger.error("Failed waiting for #{function_name} to update:\n #{e.message}")
    end
  end
```

```
    end
```

- For API details, see [UpdateFunctionCode](#) in *AWS SDK for Ruby API Reference*.

## Update function configuration

The following code example shows how to update Lambda function configuration.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
      function_name:,
      environment: {
        variables: {
          "LOG_LEVEL" => log_level
        }
      }
    })
    @lambda_client.wait_until(:function_updated_v2, { function_name: }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    rescue Aws::Lambda::Errors::ServiceException => e
      @logger.error("There was an error updating configurations for #{function_name}:\n#{e.message}")
    rescue Aws::Waiters::Errors::WaiterFailed => e
      @logger.error("Failed waiting for #{function_name} to activate:\n#{e.message}")
    end
  end
end
```

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for Ruby API Reference*.

## Scenarios

### Get started with functions

The following code example shows how to:

- Create an AWS Identity and Access Management (IAM) role that grants Lambda permission to write to logs.
- Create a Lambda function and upload handler code.

- Invoke the function with a single parameter and get results.
- Update the function code and configure its Lambda environment with an environment variable.
- Invoke the function with new parameters and get results. Display the execution log that's returned from the invocation.
- List the functions for your account.
- Delete the IAM role and the Lambda function.

For more information, see [Create a Lambda function with the console](#).

## SDK for Ruby

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Set up pre-requisite IAM permissions for a Lambda function capable of writing logs.

```
# Get an AWS Identity and Access Management (IAM) role.
#
# @param iam_role_name: The name of the role to retrieve.
# @param action: Whether to create or destroy the IAM apparatus.
# @return: The IAM role.
def manage_iam(iam_role_name, action)
  role_policy = {
    'Version': "2012-10-17",
    'Statement': [
      {
        'Effect': "Allow",
        'Principal': {
          'Service': "lambda.amazonaws.com"
        },
        'Action': "sts:AssumeRole"
      }
    ]
  }
  case action
  when "create"
    role = $iam_client.create_role(
      role_name: iam_role_name,
      assume_role_policy_document: role_policy.to_json
    )
    $iam_client.attach_role_policy(
      {
        policy_arn: "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole",
        role_name: iam_role_name
      }
    )
    $iam_client.wait_until(:role_exists, { role_name: iam_role_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    @logger.debug("Successfully created IAM role: #{role['role']['arn']}")
    @logger.debug("Enforcing a 10-second sleep to allow IAM role to activate fully.")
    sleep(10)
    return role, role_policy.to_json
  when "destroy"
    $iam_client.detach_role_policy(
      {
        policy_arn: "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole",
        role_name: iam_role_name
      }
    )
  end
end
```

```
        }
    )
    $iam_client.delete_role(
        role_name: iam_role_name
    )
    @logger.debug("Detached policy & deleted IAM role: #{iam_role_name}")
else
    raise "Incorrect action provided. Must provide 'create' or 'destroy'"
end
rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating role or attaching policy:\n#{e.message}")
end
```

Define a Lambda handler that increments a number provided as an invocation parameter.

```
require "logger"

# A function that increments a whole number by one (1) and logs the result.
# Requires a manually-provided runtime parameter, 'number', which must be Int
#
# @param event [Hash] Parameters sent when the function is invoked
# @param context [Hash] Methods and properties that provide information
# about the invocation, function, and execution environment.
# @return incremented_number [String] The incremented number.
def lambda_handler(event:, context:)
    logger = Logger.new($stdout)
    log_level = ENV["LOG_LEVEL"]
    logger.level = case log_level
                    when "debug"
                        Logger::DEBUG
                    when "info"
                        Logger::INFO
                    else
                        Logger::ERROR
                    end
    logger.debug("This is a debug log message.")
    logger.info("This is an info log message. Code executed successfully!")
    number = event["number"].to_i
    incremented_number = number + 1
    logger.info("You provided #{number.round} and it was incremented to
#{incremented_number.round}")
    incremented_number.round.to_s
end
```

Zip your Lambda function into a deployment package.

```
# Creates a Lambda deployment package in .zip format.
# This zip can be passed directly as a string to Lambda when creating the function.
#
# @param source_file: The name of the object, without suffix, for the Lambda file and
# zip.
# @return: The deployment package.
def create_deployment_package(source_file)
    Dir.chdir(File.dirname(__FILE__))
    if File.exist?("lambda_function.zip")
        File.delete("lambda_function.zip")
        @logger.debug("Deleting old zip: lambda_function.zip")
    end
    Zip::File.open("lambda_function.zip", create: true) {
        |zipfile|
```

```
        zipfile.add("lambda_function.rb", "#{source_file}.rb")
    }
    @logger.debug("Zipping #{source_file}.rb into: lambda_function.zip.")
    File.read("lambda_function.zip").to_s
rescue StandardError => e
    @logger.error("There was an error creating deployment package:\n #{e.message}")
end
```

### Create a new Lambda function.

```
# Deploys a Lambda function.
#
# @param function_name: The name of the Lambda function.
# @param handler_name: The fully qualified name of the handler function. This
#                      must include the file name and the function name.
# @param role_arn: The IAM role to use for the function.
# @param deployment_package: The deployment package that contains the function
#                            code in .zip format.
# @return: The Amazon Resource Name (ARN) of the newly created function.
def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
        role: role_arn.to_s,
        function_name:,
        handler: handler_name,
        runtime: "ruby2.7",
        code: {
            zip_file: deployment_package
        },
        environment: {
            variables: {
                "LOG_LEVEL" => "info"
            }
        }
    })
    @lambda_client.wait_until(:function_active_v2, { function_name: }) do |w|
        w.max_attempts = 5
        w.delay = 5
    end
    response
rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
end
```

### Invoke your Lambda function with optional runtime parameters.

```
# Invokes a Lambda function.
# @param function_name [String] The name of the function to invoke.
# @param payload [nil] Payload containing runtime parameters.
# @return [Object] The response from the function invocation.
def invoke_function(function_name, payload = nil)
    params = { function_name: }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
end
```

### Update your Lambda function's configuration to inject a new environment variable.

```
# Updates the environment variables for a Lambda function.
# @param function_name: The name of the function to update.
# @param log_level: The log level of the function.
# @return: Data about the update, including the status.
def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
        function_name:,
        environment: {
            variables: {
                "LOG_LEVEL" => log_level
            }
        }
    })
    @lambda_client.wait_until(:function_updated_v2, { function_name: }) do |w|
        w.max_attempts = 5
        w.delay = 5
    end
rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating configurations for #{function_name}:\n#{e.message}")
rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
end
```

Update your Lambda function's code with a different deployment package containing different code.

```
# Updates the code for a Lambda function by submitting a .zip archive that contains
# the code for the function.

# @param function_name: The name of the function to update.
# @param deployment_package: The function code to update, packaged as bytes in
#                           .zip format.
# @return: Data about the update, including the status.
def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
        function_name:,
        zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name: }) do |w|
        w.max_attempts = 5
        w.delay = 5
    end
rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for: #{function_name}:\n#{e.message}")
    nil
rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\n #{e.message}")
end
```

List all existing Lambda functions using the built-in paginator.

```
# Lists the Lambda functions for the current account.
def list_functions
    functions = []
    @lambda_client.list_functions.each do |response|
        response["functions"].each do |function|
            functions.append(function["function_name"])
        end
    end
    functions
```

```
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error executing #{function_name}:\n #{e.message}")
end
```

Delete a specific Lambda function.

```
# Deletes a Lambda function.
# @param function_name: The name of the function to delete.
def delete_function(function_name)
  print "Deleting function: #{function_name}..."
  @lambda_client.delete_function(
    function_name:
  )
  print "Done!".green
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
end
```

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.

- [CreateFunction](#)
- [DeleteFunction](#)
- [GetFunction](#)
- [Invoke](#)
- [ListFunctions](#)
- [UpdateFunctionCode](#)
- [UpdateFunctionConfiguration](#)

## Amazon S3 examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Amazon Simple Storage Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4366\)](#)
- [Scenarios \(p. 4381\)](#)

## Actions

### Add CORS rules to a bucket

The following code example shows how to add cross-origin resource sharing (CORS) rules to an S3 bucket.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
  # existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
#{e.message}"
    false
  end
end
```

- For API details, see [PutBucketCors](#) in *AWS SDK for Ruby API Reference*.

## Add a policy to a bucket

The following code example shows how to add a policy to an S3 bucket.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured with
  # an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end
```

```
# Sets a policy on a bucket.  
#  
def set_policy(policy)  
    @bucket_policy.put(policy: policy)  
    true  
rescue Aws::Errors::ServiceError => e  
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:  
#{e.message}"  
    false  
end  
end
```

- For API details, see [PutBucketPolicy in AWS SDK for Ruby API Reference](#).

## Copy an object from one bucket to another

The following code example shows how to copy an S3 object from one bucket to another.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Copy an object.

```
require "aws-sdk-s3"  
  
# Wraps Amazon S3 object actions.  
class ObjectCopyWrapper  
    attr_reader :source_object  
  
    # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is used  
    # as the source object for  
    #                                         copy actions.  
    def initialize(source_object)  
        @source_object = source_object  
    end  
  
    # Copy the source object to the specified target bucket and rename it with the target  
    # key.  
    #  
    # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the  
    # object is copied.  
    # @param target_object_key [String] The key to give the copy of the object.  
    # @return [Aws::S3::Object, nil] The copied object when successful; otherwise, nil.  
    def copy_object(target_bucket, target_object_key)  
        @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)  
        target_bucket.object(target_object_key)  
        rescue Aws::Errors::ServiceError => e  
            puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:  
#{e.message}"  
        end  
    end  
  
    # Replace the source and target bucket names with existing buckets you own and replace  
    # the source object key  
    # with an existing object in the source bucket.  
    def run_demo  
        source_bucket_name = "doc-example-bucket1"  
        source_key = "my-source-file.txt"  
        target_bucket_name = "doc-example-bucket2"
```

```
target_key = "my-target-file.txt"

source_bucket = Aws::S3::Bucket.new(source_bucket_name)
wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
target_bucket = Aws::S3::Bucket.new(target_bucket_name)
target_object = wrapper.copy_object(target_bucket, target_key)
return unless target_object

puts "Copied #{source_key} from #{source_bucket_name} to
#{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Copy an object and add server-side encryption to the destination object.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is used
  # as the source object for
  #                               copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the target
  # key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise, nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
#{e.message}"
  end
end

# Replace the source and target bucket names with existing buckets you own and replace
# the source object key
# with an existing object in the source bucket.
def run_demo
  source_bucket_name = "doc-example-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "doc-example-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key, target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
#{target_object.bucket_name}:#{target_object.key} and \"\\"
```

```
    "encrypted the target with #{target_object.server_side_encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [CopyObject](#) in *AWS SDK for Ruby API Reference*.

## Create a bucket

The following code example shows how to create an S3 bucket.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name. This
  # is a client-side object until
  #                                         create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      "None. You must create a bucket before you can get its location!"
    else
      @bucket.client.get_bucket_location(bucket: @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("doc-example-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)
```

```
    puts "Created bucket #{wrapper.bucket.name}."  
    puts "Your bucket's region is: #{wrapper.location}"  
end  
  
run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [CreateBucket](#) in *AWS SDK for Ruby API Reference*.

## Delete CORS rules from a bucket

The following code example shows how to delete CORS rules from an S3 bucket.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"  
  
# Wraps Amazon S3 bucket CORS configuration.  
class BucketCorsWrapper  
  attr_reader :bucket_cors  
  
  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an  
  # existing bucket.  
  def initialize(bucket_cors)  
    @bucket_cors = bucket_cors  
  end  
  
  # Deletes the CORS configuration of a bucket.  
  #  
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.  
  def delete_cors  
    @bucket_cors.delete  
    true  
  rescue Aws::Errors::ServiceError => e  
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:  
    #{e.message}"  
    false  
  end  
end
```

- For API details, see [DeleteBucketCors](#) in *AWS SDK for Ruby API Reference*.

## Delete a policy from a bucket

The following code example shows how to delete a policy from an S3 bucket.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Wraps an Amazon S3 bucket policy.  
class BucketPolicyWrapper
```

```
attr_reader :bucket_policy

# @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured with
# an existing bucket.
def initialize(bucket_policy)
  @bucket_policy = bucket_policy
end

def delete_policy
  @bucket_policy.delete
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's why:
#{e.message}"
  false
end

end
```

- For API details, see [DeleteBucketPolicy](#) in *AWS SDK for Ruby API Reference*.

## Delete an empty bucket

The following code example shows how to delete an empty S3 bucket.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
  answer = gets.chomp.downcase
  if answer == "y"
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- For API details, see [DeleteBucket](#) in *AWS SDK for Ruby API Reference*.

## Delete multiple objects

The following code example shows how to delete multiple objects from an S3 bucket.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.  
#  
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.  
def delete_bucket(bucket)  
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")  
    answer = gets.chomp.downcase  
    if answer == "y"  
        bucket.objects.batch_delete!  
        bucket.delete  
        puts("Emptied and deleted bucket #{bucket.name}.\n")  
    end  
rescue Aws::Errors::ServiceError => e  
    puts("Couldn't empty and delete bucket #{bucket.name}.")  
    puts("\t#{e.code}: #{e.message}")  
    raise  
end
```

- For API details, see [DeleteObjects](#) in *AWS SDK for Ruby API Reference*.

## Determine the existence and content type of an object

The following code example shows how to determine the existence and content type of an object in an S3 bucket.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"  
  
# Wraps Amazon S3 object actions.  
class ObjectExistsWrapper  
    attr_reader :object  
  
    # @param object [Aws::S3::Object] An Amazon S3 object.  
    def initialize(object)  
        @object = object  
    end  
  
    # Checks whether the object exists.  
    #  
    # @return [Boolean] True if the object exists; otherwise false.  
    def exists?  
        @object.exists?  
    rescue Aws::Errors::ServiceError => e  
        puts "Couldn't check existence of object #{@object.bucket.name}:#{@object.key}.  
Here's why: #{e.message}"  
        false  
    end  
end  
  
# Replace bucket name and object key with an existing bucket and object that you own.  
def run_demo  
    bucket_name = "doc-example-bucket"  
    object_key = "my-object.txt"  
  
    wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))  
    exists = wrapper.exists?
```

```
    puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [HeadObject](#) in *AWS SDK for Ruby API Reference*.

## Get CORS rules for a bucket

The following code example shows how to get cross-origin resource sharing (CORS) rules for an S3 bucket.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
  # existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS configuration
  # for the bucket.
  def get_cors
    @bucket_cors.data
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's why:
#{e.message}"
      nil
  end
end
```

- For API details, see [GetBucketCors](#) in *AWS SDK for Ruby API Reference*.

## Get an object from a bucket

The following code example shows how to read data from an object in an S3 bucket.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get an object.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
  # successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
    end
  end

  # Replace bucket name and object key with an existing bucket and object that you own.
  def run_demo
    bucket_name = "doc-example-bucket"
    object_key = "my-object.txt"
    target_path = "my-object-as-file.txt"

    wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
    obj_data = wrapper.get_object(target_path)
    return unless obj_data

    puts "Object #{@object_key} (#{@obj_data.content_length} bytes) downloaded to
    #{target_path}."
  end

  run_demo if $PROGRAM_NAME == __FILE__

```

Get an object and report its server-side encryption state.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
  # successful; otherwise nil.
  def get_object
    @object.get
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
    end
  end

```

```
# Replace bucket name and object key with an existing bucket and object that you own.
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? "no" :
obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [GetObject](#) in *AWS SDK for Ruby API Reference*.

## Get the policy for a bucket

The following code example shows how to get the policy for an S3 bucket.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured with
  # an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def get_policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
#{e.message}"
      nil
  end
end
```

- For API details, see [GetBucketPolicy](#) in *AWS SDK for Ruby API Reference*.

## List buckets

The following code example shows how to list S3 buckets.

## SDK for Ruby

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts "Found these buckets:"
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [ListBuckets](#) in *AWS SDK for Ruby API Reference*.

## List objects in a bucket

The following code example shows how to list objects in an S3 bucket.

## SDK for Ruby

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.

```

```
def initialize(bucket)
  @bucket = bucket
end

# Lists objects in a bucket.
#
# @param max_objects [Integer] The maximum number of objects to list.
# @return [Integer] The number of objects listed.
def list_objects(max_objects)
  count = 0
  puts "The objects in #{@bucket.name} are:"
  @bucket.objects.each do |obj|
    puts "\t#{obj.key}"
    count += 1
    break if count == max_objects
  end
  count
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list objects in bucket #{@bucket.name}. Here's why: #{e.message}"
  0
end
end

def run_demo
  bucket_name = "doc-example-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [ListObjects](#) in *AWS SDK for Ruby API Reference*.

## Set the website configuration for a bucket

The following code example shows how to set the website configuration for an S3 bucket.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object configured
  # with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the website.
  # @param error_document [String] The name of the error document to show for 4XX
  # errors.
```

```
# @return [Boolean] True when the bucket is configured as a website; otherwise, false.
def set_website(index_document, error_document)
  @bucket_website.put(
    website_configuration: {
      index_document: { suffix: index_document },
      error_document: { key: error_document }
    }
  )
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's why: #{e.message}"
  false
end

def run_demo
  bucket_name = "doc-example-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [PutBucketWebsite](#) in *AWS SDK for Ruby API Reference*.

## Upload an object to a bucket

The following code example shows how to upload an object to an S3 bucket.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Upload a file using a managed uploader (`Object.upload_file`).

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  end
end
```

```
rescue Aws::Errors::ServiceError => e
  puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
#{e.message}"
  false
end

def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Upload a file using Object.put.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, "rb") do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why: #{e.message}"
    false
  end
end

def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Upload a file using Object.put and add server-side encryption.

```
require "aws-sdk-s3"
```

```
# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name, object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
  #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- For API details, see [PutObject](#) in *AWS SDK for Ruby API Reference*.

## Scenarios

### Create a presigned URL

The following code example shows how to create a presigned URL for S3 and upload an object.

#### SDK for Ruby

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"
require "net/http"

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}."
  URI(url)
```

```
rescue Aws::Errors::ServiceError => e
    puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's why:
    #{e.message}"
end

def run_demo
    bucket_name = "doc-example-bucket"
    object_key = "my-file.txt"
    object_content = "This is the content of my-file.txt."

    bucket = Aws::S3::Bucket.new(bucket_name)
    presigned_url = get_presigned_url(bucket, object_key)
    return unless presigned_url

    response = Net::HTTP.start(presigned_url.host) do |http|
        http.send_request("PUT", presigned_url.request_uri, object_content, "content_type"
=> "")
    end

    case response
    when Net::HTTPPSuccess
        puts "Content uploaded!"
    else
        puts response.value
    end
end

run_demo if $PROGRAM_NAME == __FILE__
```

## Get started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.
- Delete a bucket.

## SDK for Ruby

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
    attr_reader :s3_resource

    # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
    def initialize(s3_resource)
        @s3_resource = s3_resource
    end
```

```
# Creates a bucket with a random name in the currently configured account and
# AWS Region.
#
# @return [Aws::S3::Bucket] The newly created bucket.
def create_bucket
  bucket = @s3_resource.create_bucket(
    bucket: "doc-example-bucket-#{Random.uuid}",
    create_bucket_configuration: {
      location_constraint: "us-east-2" # Note: only certain regions permitted
    }
  )
  puts("Created demo bucket named #{bucket.name}.")
rescue Aws::Errors::ServiceError => e
  puts("Tried and failed to create demo bucket.")
  puts("\t#{e.code}: #{e.message}")
  puts("\nCan't continue the demo without a bucket!")
  raise
else
  bucket
end

# Requests a file name from the user.
#
# @return The name of the file.
def create_file
  File.open("demo.txt", w) { |f| f.write("This is a demo file.") }
end

# Uploads a file to an Amazon S3 bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket object representing the upload
# destination
# @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded file.
def upload_file(bucket)
  File.open("demo.txt", "w+") { |f| f.write("This is a demo file.") }
  s3_object = bucket.object(File.basename("demo.txt"))
  s3_object.upload_file("demo.txt")
  puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't upload file demo.txt to #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  s3_object
end

# Downloads an Amazon S3 object to a file.
#
# @param s3_object [Aws::S3::Object] The object to download.
def download_file(s3_object)
  puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
  answer = gets.chomp.downcase
  if answer == "y"
    puts("Enter a name for the downloaded file: ")
    file_name = gets.chomp
    s3_object.download_file(file_name)
    puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't download #{s3_object.key}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

```
# Copies an Amazon S3 object to a subfolder within the same bucket.  
#  
# @param source_object [Aws::S3::Object] The source object to copy.  
# @return [Aws::S3::Object, nil] The destination object.  
def copy_object(source_object)  
  dest_object = nil  
  puts("\nDo you want to copy #{source_object.key} to a subfolder in your bucket (y/n)? ")  
  answer = gets.chomp.downcase  
  if answer == "y"  
    dest_object = source_object.bucket.object("demo-folder/#{source_object.key}")  
    dest_object.copy_from(source_object)  
    puts("Copied #{source_object.key} to #{dest_object.key}.")  
  end  
rescue Aws::Errors::ServiceError => e  
  puts("Couldn't copy #{source_object.key}.")  
  puts("\t#{e.code}: #{e.message}")  
  raise  
else  
  dest_object  
end  
  
# Lists the objects in an Amazon S3 bucket.  
#  
# @param bucket [Aws::S3::Bucket] The bucket to query.  
def list_objects(bucket)  
  puts("\nYour bucket contains the following objects:")  
  bucket.objects.each do |obj|  
    puts("\t#{obj.key}")  
  end  
rescue Aws::Errors::ServiceError => e  
  puts("Couldn't list the objects in bucket #{bucket.name}.")  
  puts("\t#{e.code}: #{e.message}")  
  raise  
end  
  
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.  
#  
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.  
def delete_bucket(bucket)  
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")  
  answer = gets.chomp.downcase  
  if answer == "y"  
    bucket.objects.batch_delete!  
    bucket.delete  
    puts("Emptied and deleted bucket #{bucket.name}.\n")  
  end  
rescue Aws::Errors::ServiceError => e  
  puts("Couldn't empty and delete bucket #{bucket.name}.")  
  puts("\t#{e.code}: #{e.message}")  
  raise  
end  
end  
  
# Runs the Amazon S3 getting started scenario.  
def run_scenario(scenario)  
  puts("-" * 88)  
  puts("Welcome to the Amazon S3 getting started demo!")  
  puts("-" * 88)  
  
  bucket = scenario.create_bucket  
  s3_object = scenario.upload_file(bucket)  
  scenario.download_file(s3_object)  
  scenario.copy_object(s3_object)  
  scenario.list_objects(bucket)  
  scenario.delete_bucket(bucket)
```

```
    puts("Thanks for watching!")
    puts("-" * 88)
rescue Aws::Errors::ServiceError
    puts("Something went wrong with the demo!")
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME ==
__FILE__
```

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

## Amazon SNS examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Amazon Simple Notification Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4385\)](#)

## Actions

### Create a topic

The following code example shows how to create an Amazon SNS topic.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-sns" # v2: require 'aws-sdk'

def topic_created?(sns_client, topic_name)

  sns_client.create_topic(name: topic_name)
  rescue StandardError => e
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
end
```

```
# Full example call:  
def run_me  
  topic_name = "TOPIC_NAME"  
  region = "REGION"  
  
  sns_client = Aws::SNS::Client.new(region: region)  
  
  puts "Creating the topic '#{topic_name}'..."  
  
  if topic_created?(sns_client, topic_name)  
    puts "The topic was created."  
  else  
    puts "The topic was not created. Stopping program."  
    exit 1  
  end  
end  
  
run_me if $PROGRAM_NAME == __FILE__
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [CreateTopic](#) in *AWS SDK for Ruby API Reference*.

### List the subscribers of a topic

The following code example shows how to retrieve the list of subscribers of an Amazon SNS topic.

#### SDK for Ruby

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-sns" # v2: require 'aws-sdk'  
  
def show_subscriptions?(sns_client, topic_arn)  
  topic = sns_client.topic(topic_arn)  
  topic.subscriptions.each do |s|  
    puts s.attributes["Endpoint"]  
  end  
  
rescue StandardError => e  
  puts "Error while sending the message: #{e.message}"  
end  
  
def run_me  
  
  topic_arn = "SNS_TOPIC_ARN"  
  region = "REGION"  
  
  sns_client = Aws::SNS::Resource.new(region: region)  
  
  puts "Listing subscriptions to the topic."  
  
  if show_subscriptions?(sns_client, topic_arn)  
  else  
    puts "There was an error. Stopping program."  
    exit 1  
  end  
end
```

```
run_me if $PROGRAM_NAME == __FILE__
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [ListSubscriptions](#) in [AWS SDK for Ruby API Reference](#).

## List topics

The following code example shows how to list Amazon SNS topics.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-sns" # v2: require 'aws-sdk'

def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
  rescue StandardError => e
    puts "Error while listing the topics: #{e.message}"
  end
end

def run_me

  region = "REGION"
  sns_client = Aws::SNS::Resource.new(region: region)

  puts "Listing the topics."

  if list_topics?(sns_client)
  else
    puts "The bucket was not created. Stopping program."
    exit 1
  end
end
run_me if $PROGRAM_NAME == __FILE__
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [ListTopics](#) in [AWS SDK for Ruby API Reference](#).

## Publish to a topic

The following code example shows how to publish messages to an Amazon SNS topic.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-sns" # v2: require 'aws-sdk'

def message_sent?(sns_client, topic_arn, message)

    sns_client.publish(topic_arn: topic_arn, message: message)
rescue StandardError => e
    puts "Error while sending the message: #{e.message}"
end

def run_me

    topic_arn = "SNS_TOPIC_ARN"
    region = "REGION"
    message = "MESSAGE" # The text of the message to send.

    sns_client = Aws::SNS::Client.new(region: region)

    puts "Message sending."

    if message_sent?(sns_client, topic_arn, message)
        puts "The message was sent."
    else
        puts "The message was not sent. Stopping program."
        exit 1
    end
end

run_me if $PROGRAM_NAME == __FILE__
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [Publish](#) in [AWS SDK for Ruby API Reference](#).

## Set topic attributes

The following code example shows how to set Amazon SNS topic attributes.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-sns" # v2: require 'aws-sdk'

policy = {
    "Version": "2008-10-17",
    "Id": "__default_policy_ID",
    "Statement": [
        {
            "Sid": "__default_statement_ID",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": ["SNS:Publish"],
            "Resource": "' + MY_TOPIC_ARN + ''",
            "Condition": {
                "ArnEquals": {
```

```
        "AWS:SourceArn":' + MY_RESOURCE_ARN + ''}
    }
}]
}'
# Replace us-west-2 with the AWS Region you're using for Amazon SNS.
sns = Aws::SNS::Resource.new(region: "REGION")

# Get topic by ARN
topic = sns.topic()

# Add policy to topic
topic.set_attributes({
    attribute_name: "POLICY_NAME",
    attribute_value: policy
})
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [SetTopicAttributes](#) in [AWS SDK for Ruby API Reference](#).

## Subscribe an email address to a topic

The following code example shows how to subscribe an email address to an Amazon SNS topic.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-sns" # v2: require 'aws-sdk'

def subscription_created?(sns_client, topic_arn, protocol, endpoint)
    sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint: endpoint)
rescue StandardError => e
    puts "Error while creating the subscription: #{e.message}"
end

# Full example call:
def run_me

    protocol = "email"
    endpoint = "EMAIL_ADDRESS"
    topic_arn = "TOPIC_ARN"
    region = "REGION"

    sns_client = Aws::SNS::Client.new(region: region)

    puts "Creating the subscription."

    if subscription_created?(sns_client, topic_arn, protocol, endpoint)
        puts "The subscription was created."
    else
        puts "The subscription was not created. Stopping program."
        exit 1
    end
end
```

```
run_me if $PROGRAM_NAME == __FILE__
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [Subscribe](#) in [AWS SDK for Ruby API Reference](#).

## AWS STS examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with AWS Security Token Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4390\)](#)

## Actions

### Assume a role

The following code example shows how to assume a role with AWS STS.

### SDK for Ruby

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Creates an AWS Security Token Service (AWS STS) client with specified credentials.  
# This is separated into a factory function so that it can be mocked for unit  
testing.  
#  
# @param key_id [String] The ID of the access key used by the STS client.  
# @param key_secret [String] The secret part of the access key used by the STS  
client.  
def create_sts_client(key_id, key_secret)  
    Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)  
end  
  
# Gets temporary credentials that can be used to assume a role.  
#  
# @param role_arn [String] The ARN of the role that is assumed when these credentials  
#                           are used.  
# @param sts_client [AWS::STS::Client] An AWS STS client.  
# @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume the  
role.  
def assume_role(role_arn, sts_client)  
    credentials = Aws::AssumeRoleCredentials.new(  
        client: sts_client,  
        role_arn: role_arn,  
        role_session_name: "create-use-assume-role-scenario"  
    )
```

```
    puts("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
end
```

- For API details, see [AssumeRole](#) in *AWS SDK for Ruby API Reference*.

## Code examples for SDK for Rust

The code examples in this topic show you how to use the AWS SDK for Rust with AWS.

### Examples

- [Single-service actions and scenarios using SDK for Rust \(p. 4391\)](#)
- [Cross-service examples using SDK for Rust \(p. 4481\)](#)

## Single-service actions and scenarios using SDK for Rust

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for Rust with AWS services.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Services

- [API Gateway examples using SDK for Rust \(p. 4392\)](#)
- [API Gateway Management API examples using SDK for Rust \(p. 4393\)](#)
- [Application Auto Scaling examples using SDK for Rust \(p. 4394\)](#)
- [AWS Batch examples using SDK for Rust \(p. 4395\)](#)
- [Amazon Cognito Identity Provider examples using SDK for Rust \(p. 4396\)](#)
- [Amazon Cognito Sync examples using SDK for Rust \(p. 4397\)](#)
- [DynamoDB examples using SDK for Rust \(p. 4398\)](#)
- [Amazon EBS examples using SDK for Rust \(p. 4403\)](#)
- [Amazon EC2 examples using SDK for Rust \(p. 4405\)](#)
- [Amazon EC2 Auto Scaling examples using SDK for Rust \(p. 4410\)](#)
- [Amazon ECR examples using SDK for Rust \(p. 4413\)](#)
- [Amazon ECS examples using SDK for Rust \(p. 4414\)](#)
- [Amazon EKS examples using SDK for Rust \(p. 4416\)](#)
- [IAM examples using SDK for Rust \(p. 4417\)](#)
- [AWS IoT examples using SDK for Rust \(p. 4435\)](#)
- [AWS KMS examples using SDK for Rust \(p. 4436\)](#)
- [Kinesis examples using SDK for Rust \(p. 4442\)](#)
- [MediaLive examples using SDK for Rust \(p. 4445\)](#)

- [MediaPackage examples using SDK for Rust \(p. 4446\)](#)
- [Amazon Polly examples using SDK for Rust \(p. 4447\)](#)
- [QLDB examples using SDK for Rust \(p. 4450\)](#)
- [Amazon RDS Data Service examples using SDK for Rust \(p. 4452\)](#)
- [Route 53 examples using SDK for Rust \(p. 4453\)](#)
- [Amazon S3 examples using SDK for Rust \(p. 4454\)](#)
- [Amazon SES API v2 examples using SDK for Rust \(p. 4468\)](#)
- [Amazon SNS examples using SDK for Rust \(p. 4472\)](#)
- [Amazon SQS examples using SDK for Rust \(p. 4475\)](#)
- [SageMaker examples using SDK for Rust \(p. 4476\)](#)
- [Secrets Manager examples using SDK for Rust \(p. 4478\)](#)
- [Systems Manager examples using SDK for Rust \(p. 4480\)](#)

## API Gateway examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon API Gateway.

**Actions** are code excerpts that show you how to call individual service functions.

**Scenarios** are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4392\)](#)

## Actions

### List REST APIs

The following code example shows how to list API Gateway REST APIs.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Displays the Amazon API Gateway REST APIs in the Region.

```
async fn show_apis(client: &Client) -> Result<(), Error> {
    let resp = client.get_rest_apis().send().await?;

    for api in resp.items().unwrap_or_default() {
        println!("ID: {}", api.id().unwrap_or_default());
        println!("Name: {}", api.name().unwrap_or_default());
        println!("Description: {}", api.description().unwrap_or_default());
        println!("Version: {}", api.version().unwrap_or_default());
    }
}
```

```
    println!(
        "Created: {}",
        api.created_date().unwrap().to_chrono_utc()
    );
    println!();
}
Ok(())
}
```

- For API details, see [GetRestApis](#) in *AWS SDK for Rust API reference*.

## API Gateway Management API examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon API Gateway Management API.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4393\)](#)

## Actions

### Send data to a connection

The following code example shows how to send data to a connection.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn send_data(
    client: &aws_sdk_apigatewaymanagement::Client,
    con_id: &str,
    data: &str,
) -> Result<(), aws_sdk_apigatewaymanagement::Error> {
    client
        .post_to_connection()
        .connection_id(con_id)
        .data(Blob::new(data))
        .send()
        .await?;

    Ok(())
}

let uri = format!(
```

```
"https://[api_id].execute-api.[region].amazonaws.com/{stage}",
api_id = api_id,
region = region,
stage = stage
)
.parse()
.expect("could not construct valid URI for endpoint");
let endpoint = Endpoint::immutable(uri);

let shared_config = aws_config::from_env().region(region_provider).load().await;
let api_management_config = config::Builder::from(&shared_config)
    .endpoint_resolver(endpoint)
    .build();
let client = Client::from_conf(api_management_config);
```

- For API details, see [PostToConnection](#) in *AWS SDK for Rust API reference*.

## Application Auto Scaling examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Application Auto Scaling.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4394\)](#)

## Actions

### Describes scaling policies

The following code example shows how to describe Application Auto Scaling scaling policies for the specified service namespace.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_policies(client: &Client) -> Result<(), Error> {
    let response = client
        .describe_scaling_policies()
        .service_namespace(ServiceNamespace::Ec2)
        .send()
        .await?;
    if let Some(policies) = response.scaling_policies() {
        println!("Auto Scaling Policies:");
        for policy in policies {
```

```
        println!("{}:{}\n", policy);
    }
    println!("Next token: {}", response.next_token());
    Ok(())
}
```

- For API details, see [DescribeScalingPolicies](#) in *AWS SDK for Rust API reference*.

## AWS Batch examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with AWS Batch.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4395\)](#)

## Actions

### Describe compute environments

The following code example shows how to describe one or more AWS Batch compute environments.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_envs(client: &Client) -> Result<(), Error> {
    let rsp = client.describe_compute_environments().send().await?;

    let compute_envs = rsp.compute_environments().unwrap_or_default();
    println!("Found {} compute environments:", compute_envs.len());
    for env in compute_envs {
        let arn = env.compute_environment_arn().unwrap_or_default();
        let name = env.compute_environment_name().unwrap_or_default();

        println!("  Name : {}", name);
        println!("  ARN:   {}", arn);
        println!();
    }
    Ok(())
}
```

- For API details, see [DescribeComputeEnvironments](#) in *AWS SDK for Rust API reference*.

## Amazon Cognito Identity Provider examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Cognito Identity Provider.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4396\)](#)

## Actions

### List the user pools

The following code example shows how to list the Amazon Cognito user pools.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client.list_user_pools().max_results(10).send().await?;
    if let Some(pools) = response.user_pools() {
        println!("User pools:");
        for pool in pools {
            println!("  ID:          {}", pool.id().unwrap_or_default());
            println!("  Name:        {}", pool.name().unwrap_or_default());
            println!("  Status:     {:?}", pool.status());
            println!("  Lambda Config:  {:?}", pool.lambda_config().unwrap());
            println!(
                "  Last modified:  {}",
                pool.last_modified_date().unwrap().to_chrono_utc()
            );
            println!(
                "  Creation date:  {:?}",
                pool.creation_date().unwrap().to_chrono_utc()
            );
            println!();
        }
    }
    println!("Next token: {}", response.next_token().unwrap_or_default());
    Ok(())
}
```

- For API details, see [ListUserPools](#) in *AWS SDK for Rust API reference*.

## Amazon Cognito Sync examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Cognito Sync.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4397\)](#)

## Actions

### List identity pools

The following code example shows how to list Amazon Cognito identity pools.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client
        .list_identity_pool_usage()
        .max_results(10)
        .send()
        .await?;

    if let Some(pools) = response.identity_pool_usages() {
        println!("Identity pools:");

        for pool in pools {
            println!(
                "  Identity pool ID: {}",
                pool.identity_pool_id().unwrap_or_default()
            );
            println!(
                "  Data storage: {}",
                pool.data_storage().unwrap_or_default()
            );
            println!(
                "  Sync sessions count: {}",
                pool.sync_sessions_count().unwrap_or_default()
            );
            println!(
                "  Last modified: {}",
                pool.last_modified_date().unwrap().to_chrono_utc()
            );
            println!();
        }
    }
}
```

```
    println!("Next token: {}", response.next_token().unwrap_or_default());  
    Ok(())  
}
```

- For API details, see [ListIdentityPoolUsage](#) in *AWS SDK for Rust API reference*.

## DynamoDB examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon DynamoDB.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4398\)](#)

## Actions

### Create a table

The following code example shows how to create a DynamoDB table.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn create_table(client: &Client, table: &str, key: &str) -> Result<(), Error> {  
    let a_name: String = key.into();  
    let table_name: String = table.into();  
  
    let ad = AttributeDefinition::builder()  
        .attribute_name(&a_name)  
        .attribute_type(ScalarAttributeType::S)  
        .build();  
  
    let ks = KeySchemaElement::builder()  
        .attribute_name(&a_name)  
        .key_type(KeyType::Hash)  
        .build();  
  
    let pt = ProvisionedThroughput::builder()  
        .read_capacity_units(10)  
        .write_capacity_units(5)  
        .build();
```

```
let create_table_response = client
    .create_table()
    .table_name(table_name)
    .key_schema(ks)
    .attribute_definitions(ad)
    .provisioned_throughput(pt)
    .send()
    .await;

match create_table_response {
    Ok(_) => {
        println!("Added table {} with key {}", table, key);
        Ok(())
    }
    Err(e) => {
        eprintln!("Got an error creating table:");
        eprintln!("{}: {}", "{}", e);
        Err(Error::Unhandled(Box::new(e)))
    }
}
```

- For API details, see [CreateTable](#) in *AWS SDK for Rust API reference*.

## Delete a table

The following code example shows how to delete a DynamoDB table.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_table(client: &Client, table: &str) -> Result<(), Error> {
    client.delete_table().table_name(table).send().await?;

    println!("Deleted table");

    Ok(())
}
```

- For API details, see [DeleteTable](#) in *AWS SDK for Rust API reference*.

## Delete an item from a table

The following code example shows how to delete an item from a DynamoDB table.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_item(
    client: &Client,
    table: &str,
    key: &str,
    value: &str,
) -> Result<(), Error> {
    match client
        .delete_item()
        .table_name(table)
        .key(key, AttributeValue::S(value.into()))
        .send()
        .await
    {
        Ok(_) => {
            println!("Deleted item from table");
            Ok(())
        }
        Err(e) => Err(Error::Unhandled(Box::new(e))),
    }
}
```

- For API details, see [DeleteItem](#) in *AWS SDK for Rust API reference*.

## List tables

The following code example shows how to list DynamoDB tables.

### SDK for Rust

**Note**

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_tables(client: &Client) -> Result<(), Error> {
    let paginator = client.list_tables().intoPaginator().items().send();
    let table_names = paginator.collect::<Result<Vec<_>, _>>().await?;

    println!("Tables:");

    for name in &table_names {
        println!("  {}", name);
    }

    println!("Found {} tables", table_names.len());
    Ok(())
}
```

Determine whether table exists.

```
pub async fn table_exists(client: &Client, table: &str) -> Result<bool, Error> {
    debug!("Checking for table: {table}");
    let table_list = client.list_tables().send().await;

    match table_list {
        Ok(list) => Ok(list.table_names().as_ref().unwrap().contains(&table.into())),
        Err(e) => Err(Error::Unhandled(Box::new(e))),
    }
}
```

- For API details, see [ListTables](#) in *AWS SDK for Rust API reference*.

## Put an item in a table

The following code example shows how to put an item in a DynamoDB table.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn add_item(client: &Client, item: Item, table: &String) -> Result<(), Error> {
    let user_av = AttributeValue::S(item.username);
    let type_av = AttributeValue::S(item.p_type);
    let age_av = AttributeValue::S(item.age);
    let first_av = AttributeValue::S(item.first);
    let last_av = AttributeValue::S(item.last);

    let request = client
        .put_item()
        .table_name(table)
        .item("username", user_av)
        .item("account_type", type_av)
        .item("age", age_av)
        .item("first_name", first_av)
        .item("last_name", last_av);

    println!("Executing request [{request:?}] to add item...");

    let resp = request.send().await?;

    let attributes = resp.attributes().unwrap();

    println!(
        "Added user {:?}, {:?} {:?}, age {:?} as {:?} user",
        attributes.get("username"),
        attributes.get("first_name"),
        attributes.get("last_name"),
        attributes.get("age"),
        attributes.get("p_type")
    );
}

Ok(())
}
```

- For API details, see [PutItem](#) in *AWS SDK for Rust API reference*.

## Query a table

The following code example shows how to query a DynamoDB table.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Find the movies made in the specified year.

```
pub async fn movies_in_year(
    client: &Client,
    table_name: &str,
    year: u16,
) -> Result<Vec<Movie>, MovieError> {
    let results = client
        .query()
        .table_name(table_name)
        .key_condition_expression("#yr = :yyyy")
        .expression_attribute_names("#yr", "year")
        .expression_attribute_values(":yyyy", AttributeValue::N(year.to_string()))
        .send()
        .await?;

    if let Some(items) = results.items {
        let movies = items.iter().map(|v| v.into()).collect();
        Ok(movies)
    } else {
        Ok(vec![])
    }
}
```

- For API details, see [Query](#) in *AWS SDK for Rust API reference*.

## Scan a table

The following code example shows how to scan a DynamoDB table.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_items(client: &Client, table: &str) -> Result<(), Error> {
```

```
let items: Result<Vec<_>, _> = client
    .scan()
    .table_name(table)
    .into_paginator()
    .items()
    .send()
    .collect()
    .await;

    println!("Items in table:");
    for item in items? {
        println!("  {:?}", item);
    }

    Ok(())
}
```

- For API details, see [Scan](#) in *AWS SDK for Rust API reference*.

## Amazon EBS examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Elastic Block Store.

**Actions** are code excerpts that show you how to call individual service functions.

**Scenarios** are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4403\)](#)

## Actions

### Create a snapshot

The following code example shows how to create an Amazon EBS snapshot.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn start(client: &Client, description: &str) -> Result<String, Error> {
    let snapshot = client
        .start_snapshot()
        .description(description)
        .encrypted(false)
        .volume_size(1)
        .send()
        .await?;
```

```
    Ok(snapshot.snapshot_id.unwrap())
}
```

- For API details, see [StartSnapshot](#) in *AWS SDK for Rust API reference*.

## Seal and complete a snapshot

The following code example shows how to seal and complete an Amazon EBS snapshot.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn finish(client: &Client, id: &str) -> Result<(), Error> {
    client
        .complete_snapshot()
        .changed_blocks_count(2)
        .snapshot_id(id)
        .send()
        .await?;

    println!("Snapshot ID {}", id);
    println!("The state is 'completed' when all of the modified blocks have been transferred to Amazon S3.");
    println!("Use the get-snapshot-state code example to get the state of the snapshot.");

    Ok(())
}
```

- For API details, see [CompleteSnapshot](#) in *AWS SDK for Rust API reference*.

## Write a block of data to a snapshot

The following code example shows how to write a block of data to an Amazon EBS snapshot.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn add_block(
    client: &Client,
```

```
        id: &str,
        idx: usize,
        block: Vec<u8>,
        checksum: &str,
    ) -> Result<(), Error> {
    client
        .put_snapshot_block()
        .snapshot_id(id)
        .block_index(idx as i32)
        .block_data(ByteStream::from(block))
        .checksum(checksum)
        .checksum_algorithm(ChecksumAlgorithm::ChecksumAlgorithmSha256)
        .data_length(EBS_BLOCK_SIZE as i32)
        .send()
        .await?;

    Ok(())
}
```

- For API details, see [PutSnapshotBlock](#) in *AWS SDK for Rust API reference*.

## Amazon EC2 examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Elastic Compute Cloud.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4405\)](#)

## Actions

### Delete a snapshot

The following code example shows how to delete an Amazon EBS snapshot.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn delete_snapshot(client: &Client, id: &str) -> Result<(), Error> {
    client.delete_snapshot().snapshot_id(id).send().await?;

    println!("Deleted");

    Ok(())
}
```

```
}
```

- For API details, see [DeleteSnapshot in AWS SDK for Rust API reference](#).

## Describe Regions

The following code example shows how to describe Amazon EC2 Regions.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_regions(client: &Client) -> Result<(), Error> {
    let rsp = client.describe_regions().send().await?;

    println!("Regions:");
    for region in rsp.regions().unwrap_or_default() {
        println!("  {}", region.region_name().unwrap());
    }
    Ok(())
}
```

- For API details, see [DescribeRegions in AWS SDK for Rust API reference](#).

## Describe instance status

The following code example shows how to describe the status of Amazon EC2 instances.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_all_events(client: &Client) -> Result<(), Error> {
    let resp = client.describe_regions().send().await.unwrap();

    for region in resp.regions.unwrap_or_default() {
        let reg: &'static str = Box::leak(Box::from(region.region_name().unwrap()));
        let region_provider = RegionProviderChain::default_provider().or_else(reg);
        let config = aws_config::from_env().region(region_provider).load().await;
        let new_client = Client::new(&config);

        let resp = new_client.describe_instance_status().send().await;
```

```
    println!("Instances in region {}:", reg);
    println!();

    for status in resp.unwrap().instance_statuses().unwrap_or_default() {
        println!(
            "    Events scheduled for instance ID: {}",
            status.instance_id().unwrap_or_default()
        );
        for event in status.events().unwrap_or_default() {
            println!("        Event ID:      {}", event.instance_event_id().unwrap());
            println!("        Description: {}", event.description().unwrap());
            println!("        Event code:   {}", event.code().unwrap().as_ref());
            println!();
        }
    }
    Ok(())
}
```

- For API details, see [DescribeInstanceStatus](#) in *AWS SDK for Rust API reference*.

## Describe instances

The following code example shows how to describe Amazon EC2 instances.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_state(client: &Client, ids: Option<Vec<String>>) -> Result<(), Error> {
    let resp = client
        .describe_instances()
        .set_instance_ids(ids)
        .send()
        .await?;

    for reservation in resp.reservations().unwrap_or_default() {
        for instance in reservation.instances().unwrap_or_default() {
            println!("Instance ID: {}", instance.instance_id().unwrap());
            println!("State:      {:?}", instance.state().unwrap().name().unwrap());
        }
        println!();
    }
    Ok(())
}
```

- For API details, see [DescribeInstances](#) in *AWS SDK for Rust API reference*.

## Describe snapshots

The following code example shows how to describe Amazon EBS snapshots.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Shows the state of a snapshot.

```
async fn show_state(client: &Client, id: &str) -> Result<(), Error> {
    let resp = client
        .describe_snapshots()
        .filters(Filter::builder().name("snapshot-id").values(id).build())
        .send()
        .await?;

    println!(
        "State: {}",
        resp.snapshots()
            .unwrap()
            .first()
            .unwrap()
            .state()
            .unwrap()
            .as_ref()
    );
    Ok(())
}
```

```
async fn show_snapshots(client: &Client) -> Result<(), Error> {
    // "self" represents your account ID.
    // You can list the snapshots for any account by replacing
    // "self" with that account ID.
    let resp = client.describe_snapshots().owner_ids("self").send().await?;
    let snapshots = resp.snapshots().unwrap();
    let length = snapshots.len();

    for snapshot in snapshots {
        println!(
            "ID:          {}",
            snapshot.snapshot_id().unwrap_or_default()
        );
        println!(
            "Description: {}",
            snapshot.description().unwrap_or_default()
        );
        println!("State:      {}", snapshot.state().unwrap().as_ref());
        println!();
    }

    println!();
    println!("Found {} snapshot(s)", length);
    println!();
}
```

```
    Ok(())
}
```

- For API details, see [DescribeSnapshots](#) in *AWS SDK for Rust API reference*.

## Enable monitoring

The following code example shows how to enable monitoring for a running Amazon EC2 instance.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn enable_monitoring(client: &Client, id: &str) -> Result<(), Error> {
    client.monitor_instances().instance_ids(id).send().await?;

    println!("Enabled monitoring");

    Ok(())
}
```

- For API details, see [MonitorInstances](#) in *AWS SDK for Rust API reference*.

## Reboot an instance

The following code example shows how to reboot an Amazon EC2 instance.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn reboot_instance(client: &Client, id: &str) -> Result<(), Error> {
    client.reboot_instances().instance_ids(id).send().await?;

    println!("Rebooted instance.");
    Ok(())
}
```

- For API details, see [RebootInstances](#) in *AWS SDK for Rust API reference*.

## Start an instance

The following code example shows how to start an Amazon EC2 instance.

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn start_instance(client: &Client, id: &str) -> Result<(), Error> {
    client.start_instances().instance_ids(id).send().await?;

    println!("Started instance.");

    Ok(())
}
```

- For API details, see [StartInstances](#) in *AWS SDK for Rust API reference*.

## Stop an instance

The following code example shows how to stop an Amazon EC2 instance.

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn stop_instance(client: &Client, id: &str) -> Result<(), Error> {
    client.stop_instances().instance_ids(id).send().await?;

    println!("Stopped instance.");

    Ok(())
}
```

- For API details, see [StopInstances](#) in *AWS SDK for Rust API reference*.

## Amazon EC2 Auto Scaling examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon EC2 Auto Scaling.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4411\)](#)

## Actions

### Create a group

The following code example shows how to create an Auto Scaling group.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn create_group(client: &Client, name: &str, id: &str) -> Result<(), Error> {
    client
        .create_auto_scaling_group()
        .auto_scaling_group_name(name)
        .instance_id(id)
        .min_size(1)
        .max_size(5)
        .send()
        .await?;

    println!("Created AutoScaling group");

    Ok(())
}
```

- For API details, see [CreateAutoScalingGroup](#) in *AWS SDK for Rust API reference*.

### Delete a group

The following code example shows how to delete an Auto Scaling group.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn delete_group(client: &Client, name: &str, force: bool) -> Result<(), Error> {
    client
        .delete_auto_scaling_group()
        .auto_scaling_group_name(name)
        .set_force_delete(if force { Some(true) } else { None })
        .send()
        .await?;

    println!("Deleted Auto Scaling group");

    Ok(())
}
```

- For API details, see [DeleteAutoScalingGroup](#) in *AWS SDK for Rust API reference*.

## Get information about groups

The following code example shows how to get information about Auto Scaling groups.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups().unwrap_or_default();

    for group in groups {
        println!("  {}", group.auto_scaling_group_name().unwrap_or_default());
        println!(
            "    ARN:      {}",
            group.auto_scaling_group_arn().unwrap_or_default()
        );
        println!("    Minimum size: {}", group.min_size().unwrap_or_default());
        println!("    Maximum size: {}", group.max_size().unwrap_or_default());
        println!();
    }

    println!("Found {} group(s)", groups.len());
}

Ok(())
}
```

- For API details, see [DescribeAutoScalingGroups](#) in *AWS SDK for Rust API reference*.

## Update a group

The following code example shows how to update the configuration for an Auto Scaling group.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn update_group(client: &Client, name: &str, size: i32) -> Result<(), Error> {
    client
        .update_auto_scaling_group()
        .auto_scaling_group_name(name)
        .max_size(size)
}
```

```
.send()  
.await?;  
  
println!("Updated AutoScaling group");  
  
Ok(())
}
```

- For API details, see [UpdateAutoScalingGroup](#) in *AWS SDK for Rust API reference*.

## Amazon ECR examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Elastic Container Registry.

**Actions** are code excerpts that show you how to call individual service functions.

**Scenarios** are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4413\)](#)

## Actions

### List the image IDs

The following code example shows how to list the image IDs for a repository.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_images(  
    client: &aws_sdk_ecr::Client,  
    repository: &str,  
) -> Result<(), aws_sdk_ecr::Error> {  
    let rsp = client  
        .list_images()  
        .repository_name(repository)  
        .send()  
        .await?;  
  
    let images = rsp.image_ids().unwrap_or_default();  
  
    println!("found {} images", images.len());  
  
    for image in images {  
        println!(  
            "image: {}:{}",  
            image.image_tag().unwrap(),  
            image.image_digest().unwrap()  
        );  
    }  
}
```

```
        }  
  
    Ok(())  
}
```

- For API details, see [ListImages](#) in *AWS SDK for Rust API reference*.

## List your repositories

The following code example shows how to list your repositories.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_repos(client: &aws_sdk_ecr::Client) -> Result<(), aws_sdk_ecr::Error> {  
    let rsp = client.describe.repositories().send().await?;  
  
    let repos = rsp.repositories().unwrap_or_default();  
  
    println!("Found {} repositories:", repos.len());  
  
    for repo in repos {  
        println!("  ARN: {}", repo.repository_arn().unwrap());  
        println!("  Name: {}", repo.repository_name().unwrap());  
    }  
  
    Ok(())  
}
```

- For API details, see [DescribeRepositories](#) in *AWS SDK for Rust API reference*.

## Amazon ECS examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Elastic Container Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4414\)](#)

## Actions

### Create a cluster

The following code example shows how to create an Amazon ECS cluster.

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_cluster(client: &aws_sdk_ecs::Client, name: &str) -> Result<(),  
aws_sdk_ecs::Error> {  
    let cluster = client.create_cluster().cluster_name(name).send().await?;  
    println!("cluster created: {:?}", cluster);  
  
    Ok(())  
}
```

- For API details, see [CreateCluster](#) in *AWS SDK for Rust API reference*.

## Delete a cluster

The following code example shows how to delete an Amazon ECS cluster.

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn remove_cluster(  
    client: &aws_sdk_ecs::Client,  
    name: &str,  
) -> Result<(), aws_sdk_ecs::Error> {  
    let cluster_deleted = client.delete_cluster().cluster(name).send().await?;  
    println!("cluster deleted: {:?}", cluster_deleted);  
  
    Ok(())  
}
```

- For API details, see [DeleteCluster](#) in *AWS SDK for Rust API reference*.

## List your clusters

The following code example shows how to list your Amazon ECS clusters.

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_clusters(client: &aws_sdk_ecs::Client) -> Result<(), aws_sdk_ecs::Error> {
    let resp = client.describe_clusters().send().await?;

    let clusters = resp.clusters().unwrap_or_default();
    println!("Found {} clusters:", clusters.len());

    for cluster in clusters {
        println!("  ARN: {}", cluster.cluster_arn().unwrap());
        println!("  Name: {}", cluster.cluster_name().unwrap());
    }
    Ok(())
}
```

- For API details, see [DescribeClusters in AWS SDK for Rust API reference](#).

## Amazon EKS examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Elastic Kubernetes Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

**Topics**

- [Actions \(p. 4416\)](#)

### Actions

#### Create a cluster control plane

The following code example shows how to create an Amazon EKS cluster control plane.

**SDK for Rust**

**Note**

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_cluster(
    client: &aws_sdk_eks::Client,
    name: &str,
    arn: &str,
    subnet_ids: Vec<String>,
) -> Result<(), aws_sdk_eks::Error> {
    let cluster = client
```

```
.create_cluster()
.name(name)
.role_arn(arn)
.resources_vpc_config(
    VpcConfigRequest::builder()
        .set_subnet_ids(Some(subnet_ids))
        .build(),
)
.send()
.await?;
println!("cluster created: {:?}", cluster);

Ok(())
}
```

- For API details, see [CreateCluster](#) in *AWS SDK for Rust API reference*.

## Delete a cluster control plane

The following code example shows how to delete an Amazon EKS cluster.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn remove_cluster(
    client: &aws_sdk_eks::Client,
    name: &str,
) -> Result<(), aws_sdk_eks::Error> {
    let cluster_deleted = client.delete_cluster().name(name).send().await?;
    println!("cluster deleted: {:?}", cluster_deleted);

    Ok(())
}
```

- For API details, see [DeleteCluster](#) in *AWS SDK for Rust API reference*.

## IAM examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with AWS Identity and Access Management.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4418\)](#)
- [Scenarios \(p. 4431\)](#)

## Actions

### Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn attach_role_policy(
    client: &iamClient,
    role: &Role,
    policy: &Policy,
) -> Result<AttachRolePolicyOutput, SdkError<AttachRolePolicyError>> {
    client
        .attach_role_policy()
        .role_name(role.role_name.as_ref().unwrap())
        .policy_arn(policy.arn.as_ref().unwrap())
        .send()
        .await
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Rust API reference*.

### Attach a policy to a user

The following code example shows how to attach an IAM policy to a user.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn attach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .attach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- For API details, see [AttachUserPolicy](#) in *AWS SDK for Rust API reference*.

## Create a policy

The following code example shows how to create an IAM policy.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn create_policy(
    client: &iamClient,
    policy_name: &str,
    policy_document: &str,
) -> Result<Policy, iamError> {
    let policy = client
        .create_policy()
        .policy_name(policy_name)
        .policy_document(policy_document)
        .send()
        .await?;
    Ok(policy.policy.unwrap())
}
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Rust API reference*.

## Create a role

The following code example shows how to create an IAM role.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn create_role(
    client: &iamClient,
    role_name: &str,
    role_policy_document: &str,
) -> Result<Role, iamError> {
    let response: CreateRoleOutput = loop {
        if let Ok(response) = client
            .create_role()
            .role_name(role_name)
            .assume_role_policy_document(role_policy_document)
            .send()
            .await
        {
            break response;
        }
    }
}
```

```
        }
    };

    Ok(response.role.unwrap())
}
```

- For API details, see [CreateRole](#) in *AWS SDK for Rust API reference*.

## Create a service-linked role

The following code example shows how to create an IAM service-linked role.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn create_service_linked_role(
    client: &iamClient,
    aws_service_name: String,
    custom_suffix: Option<String>,
    description: Option<String>,
) -> Result<CreateServiceLinkedRoleOutput, SdkError<CreateServiceLinkedRoleError>> {
    let response = client
        .create_service_linked_role()
        .aws_service_name(aws_service_name)
        .set_custom_suffix(custom_suffix)
        .set_description(description)
        .send()
        .await?;

    Ok(response)
}
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for Rust API reference*.

## Create a user

The following code example shows how to create an IAM user.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn create_user(client: &iamClient, user_name: &str) -> Result<User, iamError> {
    let response = client.create_user().user_name(user_name).send().await;
```

```
    Ok(response.user.unwrap())
}
```

- For API details, see [CreateUser](#) in *AWS SDK for Rust API reference*.

## Create an access key

The following code example shows how to create an IAM access key.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn create_access_key(client: &iamClient, user_name: &str) ->
Result<AccessKey, iamError> {
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<CreateAccessKeyOutput, SdkError<CreateAccessKeyError>> = loop
    {
        match client.create_access_key().user_name(user_name).send().await {
            Ok(inner_response) => {
                break Ok(inner_response);
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
                    break Err(e);
                }
                sleep(Duration::from_secs(2)).await;
            }
        };
    }

    Ok(response.unwrap().access_key.unwrap())
}
```

- For API details, see [CreateAccessKey](#) in *AWS SDK for Rust API reference*.

## Delete a policy

The following code example shows how to delete an IAM policy.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_policy(client: &iamClient, policy: Policy) -> Result<(), iamError> {
    client
        .delete_policy()
        .policy_arn(policy.arn.unwrap())
        .send()
        .await?;
    Ok(())
}
```

- For API details, see [DeletePolicy](#) in *AWS SDK for Rust API reference*.

## Delete a role

The following code example shows how to delete an IAM role.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_role(client: &iamClient, role: &Role) -> Result<(), iamError> {
    let role = role.clone();
    loop {
        match client
            .delete_role()
            .role_name(role.role_name.as_ref().unwrap())
            .send()
            .await
        {
            Ok(_) => {
                break;
            }
            Err(_) => {
                sleep(Duration::from_secs(2)).await;
            }
        }
    }
    Ok(())
}
```

- For API details, see [DeleteRole](#) in *AWS SDK for Rust API reference*.

## Delete a service-linked role

The following code example shows how to delete an IAM service-linked role.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_service_linked_role(
    client: &iamClient,
    role_name: &str,
) -> Result<(), iamError> {
    client
        .delete_service_linked_role()
        .role_name(role_name)
        .send()
        .await?;

    Ok(())
}
```

- For API details, see [DeleteServiceLinkedRole](#) in *AWS SDK for Rust API reference*.

## Delete a user

The following code example shows how to delete an IAM user.

### SDK for Rust

**Note**

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_user(client: &iamClient, user: &User) -> Result<(), SdkError<DeleteUserError>> {
    let user = user.clone();
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<(), SdkError<DeleteUserError>> = loop {
        match client
            .delete_user()
            .user_name(user.user_name.as_ref().unwrap())
            .send()
            .await
        {
            Ok(_) => {
                break Ok(());
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
                    break Err(e);
                }
                sleep(Duration::from_secs(2)).await;
            }
        }
    };
}
```

```
    response  
}
```

- For API details, see [DeleteUser](#) in *AWS SDK for Rust API reference*.

## Delete an access key

The following code example shows how to delete an IAM access key.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_access_key(  
    client: &iamClient,  
    user: &User,  
    key: &AccessKey,  
) -> Result<(), iamError> {  
    loop {  
        match client  
            .delete_access_key()  
            .user_name(user.user_name.as_ref().unwrap())  
            .access_key_id(key.access_key_id.as_ref().unwrap())  
            .send()  
            .await  
        {  
            Ok(_) => {  
                break;  
            }  
            Err(e) => {  
                println!("Can't delete the access key: {:?}", e);  
                sleep(Duration::from_secs(2)).await;  
            }  
        }  
        Ok(())  
    }  
}
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for Rust API reference*.

## Delete an inline policy from a user

The following code example shows how to delete an inline IAM policy from a user.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_user_policy(
    client: &iamClient,
    user: &User,
    policy_name: &str,
) -> Result<(), SdkError<DeleteUserPolicyError>> {
    client
        .delete_user_policy()
        .user_name(user.user_name.as_ref().unwrap())
        .policy_name(policy_name)
        .send()
        .await?;

    Ok(())
}
```

- For API details, see [DeleteUserPolicy](#) in *AWS SDK for Rust API reference*.

## Detach a policy from a role

The following code example shows how to detach an IAM policy from a role.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn detach_role_policy(
    client: &iamClient,
    role_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_role_policy()
        .role_name(role_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- For API details, see [DetachRolePolicy](#) in *AWS SDK for Rust API reference*.

## Detach a policy from a user

The following code example shows how to detach an IAM policy from a user.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn detach_user_policy(  
    client: &iamClient,  
    user_name: &str,  
    policy_arn: &str,  
) -> Result<(), iamError> {  
    client  
        .detach_user_policy()  
        .user_name(user_name)  
        .policy_arn(policy_arn)  
        .send()  
        .await?  
  
    Ok(())  
}
```

- For API details, see [DetachUserPolicy](#) in *AWS SDK for Rust API reference*.

## Get a role

The following code example shows how to get an IAM role.

### SDK for Rust

**Note**

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn get_role(  
    client: &iamClient,  
    role_name: String,  
) -> Result<GetRoleOutput, SdkError<GetRoleError>> {  
    let response = client.get_role().role_name(role_name).send().await?  
    Ok(response)  
}
```

- For API details, see [GetRole](#) in *AWS SDK for Rust API reference*.

## Get the account password policy

The following code example shows how to get the IAM account password policy.

### SDK for Rust

**Note**

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn get_account_password_policy(  
    client: &iamClient,  
) -> Result<GetAccountPasswordPolicyOutput, SdkError<GetAccountPasswordPolicyError>> {  
    let response = client.get_account_password_policy().send().await?;  
  
    Ok(response)  
}
```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for Rust API reference*.

## List SAML providers

The following code example shows how to list SAML providers for IAM.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_saml_providers(  
    client: &Client,  
) -> Result<ListSamlProvidersOutput, SdkError<ListSAMLProvidersError>> {  
    let response = client.list_saml_providers().send().await?;  
  
    Ok(response)  
}
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for Rust API reference*.

## List groups

The following code example shows how to list IAM groups.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_groups(  
    client: &iamClient,  
    path_prefix: Option<String>,  
    marker: Option<String>,  
    max_items: Option<i32>,  
) -> Result<ListGroupsOutput, SdkError<ListGroupsError>> {  
    let response = client
```

```
.list_groups()  
.set_path_prefix(path_prefix)  
.set_marker(marker)  
.set_max_items(max_items)  
.send()  
.await?;  
  
Ok(response)  
}
```

- For API details, see [ListGroups](#) in *AWS SDK for Rust API reference*.

## List inline policies for a role

The following code example shows how to list inline policies for an IAM role.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_role_policies(  
    client: &iamClient,  
    role_name: &str,  
    marker: Option<String>,  
    max_items: Option<i32>,  
) -> Result<ListRolePoliciesOutput, SdkError<ListRolePoliciesError>> {  
    let response = client  
        .list_role_policies()  
        .role_name(role_name)  
        .set_marker(marker)  
        .set_max_items(max_items)  
        .send()  
        .await?;  
  
    Ok(response)  
}
```

- For API details, see [ListRolePolicies](#) in *AWS SDK for Rust API reference*.

## List policies

The following code example shows how to list IAM policies.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_policies(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListPoliciesOutput, SdkError<ListPoliciesError>> {
    let response = client
        .list_policies()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- For API details, see [ListPolicies](#) in *AWS SDK for Rust API reference*.

## List policies attached to a role

The following code example shows how to list policies attached to an IAM role.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_attached_role_policies(
    client: &iamClient,
    role_name: String,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListAttachedRolePoliciesOutput, SdkError<ListAttachedRolePoliciesError>> {
    let response = client
        .list_attached_role_policies()
        .role_name(role_name)
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for Rust API reference*.

## List roles

The following code example shows how to list IAM roles.

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_roles(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolesOutput, SdkError<ListRolesError>> {
    let response = client
        .list_roles()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- For API details, see [ListRoles](#) in *AWS SDK for Rust API reference*.

## List users

The following code example shows how to list IAM users.

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_users(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListUsersOutput, SdkError<ListUsersError>> {
    let response = client
        .list_users()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Rust API reference*.

## Scenarios

### Create a user and assume a role

The following code example shows how to:

- Create a user who has no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list Amazon S3 buckets using temporary credentials.
- Delete the policy, role, and user.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{Client as iamClient, Credentials as iamCredentials};
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use aws_types::region::Region;
use std::borrow::Borrow;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document) =
        initialize_variables().await;

    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{}:?", e);
    }

    Ok(())
}

async fn initialize_variables() -> (iamClient, String, String, String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-west-2"));

    let shared_config = aws_config::from_env().region(region_provider).load().await;
```

```

let client = iamClient::new(&shared_config);
let uuid = Uuid::new_v4().to_string();

let list_all_buckets_policy_document = "{"
    \\"Version\\": \\"2012-10-17\",
    \\"Statement\\": [
        \\"Effect\\": \\"Allow\",
        \\"Action\\": \"s3>ListAllMyBuckets\",
        \\"Resource\\": \"arn:aws:s3:::*\"]
}
.to_string();
let inline_policy_document = "{"
    \\"Version\\": \\"2012-10-17\",
    \\"Statement\\": [
        \\"Effect\\": \\"Allow\",
        \\"Action\\": \"sts:AssumeRole\",
        \\"Resource\\": \"{}\"]
}
.to_string();

(
    client,
    uuid,
    list_all_buckets_policy_document,
    inline_policy_document,
)
}

async fn run_iam_operations(
    client: iamClient,
    uuid: String,
    list_all_buckets_policy_document: String,
    inline_policy_document: String,
) -> Result<(), iamError> {
    let user = iam_service::create_user(&client, &format!("{}{}", "iam_demo_user_", uuid)).await?;
    println!(
        "Created the user with the name: {}",
        user.user_name.as_ref().unwrap()
    );
    let key = iam_service::create_access_key(&client,
user.user_name.as_ref().unwrap()).await?;

    let assume_role_policy_document = "{"
        \\"Version\\": \\"2012-10-17\",
        \\"Statement\\": [
            \\"Effect\\": \\"Allow\",
            \\"Principal\\": \\"AWS\": \"{}\",
            \\"Action\\": \"sts:AssumeRole\"
        ]
    }
.to_string()
.replace("{}", user.arn.as_ref().unwrap());

    let assume_role_role = iam_service::create_role(
        &client,
        &format!("{}{}", "iam_demo_role_", uuid),
        &assume_role_policy_document,
    )
.await?;
    println!(
        "Created the role with the ARN: {}",
        assume_role_role.arn.as_ref().unwrap()
    );

    let list_all_buckets_policy = iam_service::create_policy(

```

```
    &client,
    &format!("{}{}", "iam_demo_policy_", uuid),
    &list_all_buckets_policy_document,
)
.await?;
println!(
    "Created policy: {}",
    list_all_buckets_policy.policy_name.as_ref().unwrap()
);

let attach_role_policy_result =
    iam_service::attach_role_policy(&client, &assume_role_role,
&list_all_buckets_policy)
    .await?;
println!(
    "Attached the policy to the role: {:?}", attach_role_policy_result
);

let inline_policy_name = format!("{}{}", "iam_demo_inline_policy_", uuid);
let inline_policy_document =
    inline_policy_document.replace("{}", assume_role_role.arn.as_ref().unwrap());
iam_service::create_user_policy(&client, &user, &inline_policy_name,
&inline_policy_document)
    .await?;
println!("Created inline policy.");

//First, fail to list the buckets with the user.
let creds = iamCredentials::from_keys(
    key.access_key_id.as_ref().unwrap(),
    key.secret_access_key.as_ref().unwrap(),
    None,
);
let fail_config = aws_config::from_env()
    .credentials_provider(creds.clone())
    .load()
    .await;
println!("Fail config: {:?}", fail_config);
let fail_client: s3Client = s3Client::new(&fail_config);
match fail_client.list_buckets().send().await {
    Ok(e) => {
        println!("This should not run. {:?}", e);
    }
    Err(e) => {
        println!("Successfully failed with error: {:?}", e)
    }
}

let sts_config = aws_config::from_env()
    .credentials_provider(creds.clone())
    .load()
    .await;
let sts_client: stsClient = stsClient::new(&sts_config);
sleep(Duration::from_secs(10)).await;
let assumed_role = sts_client
    .assume_role()
    .role_arn(assume_role_role.arn.as_ref().unwrap())
    .role_session_name(&format!("{}{}", "iam_demo_assumerole_session_", uuid))
    .send()
    .await;
println!("Assumed role: {:?}", assumed_role);
sleep(Duration::from_secs(10)).await;

let assumed_credentials = iamCredentials::from_keys(
    assumed_role
        .as_ref()

```

```
.unwrap()
.credentials
.as_ref()
.unwrap()
.access_key_id
.as_ref()
.unwrap(),
assumed_role
.as_ref()
.unwrap()
.credentials
.as_ref()
.unwrap()
.secret_access_key
.as_ref()
.unwrap(),
assumed_role
.as_ref()
.unwrap()
.credentials
.as_ref()
.unwrap()
.session_token
.borrow()
.clone(),
);

let succeed_config = aws_config::from_env()
.credentials_provider(assumed_credentials)
.load()
.await;
println!("succeed config: {:?}", succeed_config);
let succeed_client: s3Client = s3Client::new(&succeed_config);
sleep(Duration::from_secs(10)).await;
match succeed_client.list_buckets().send().await {
    Ok(_) => {
        println!("This should now run successfully.")
    }
    Err(e) => {
        println!("This should not run. {:?}", e);
        panic!()
    }
}
//Clean up.
iam_service::detach_role_policy(
    &client,
    assume_role.role_name.as_ref().unwrap(),
    list_all_buckets_policy.arn.as_ref().unwrap(),
)
.await?;
iam_service::delete_policy(&client, list_all_buckets_policy).await?;
iam_service::delete_role(&client, &assume_role.role_name).await?;
println!(
    "Deleted role {}",
    assume_role.role_name.as_ref().unwrap()
);
iam_service::delete_access_key(&client, &user, &key).await?;
println!("Deleted key for {}", key.user_name.as_ref().unwrap());
iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
println!("Deleted inline user policy: {}", inline_policy_name);
iam_service::delete_user(&client, &user).await?;
println!("Deleted user {}", user.user_name.as_ref().unwrap());

Ok(())
}
```

- For API details, see the following topics in *AWS SDK for Rust API reference*.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)
  - [DeletePolicy](#)
  - [DeleteRole](#)
  - [DeleteUser](#)
  - [DeleteUserPolicy](#)
  - [DetachRolePolicy](#)
  - [PutUserPolicy](#)

## AWS IoT examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with AWS IoT.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4435\)](#)

## Actions

### Get endpoint information

The following code example shows how to get AWS IoT endpoint information.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_address(client: &Client, endpoint_type: &str) -> Result<(), Error> {
    let resp = client
        .describe_endpoint()
        .endpoint_type(endpoint_type)
        .send()
        .await?;
```

```
    println!("Endpoint address: {}", resp.endpoint_address.unwrap());  
  
    println!();  
  
    Ok(())
}
```

- For API details, see [DescribeEndpoint](#) in *AWS SDK for Rust API reference*.

## List your things

The following code example shows how to list your AWS IoT things.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_things(client: &Client) -> Result<(), Error> {
    let resp = client.list_things().send().await?;

    println!("Things:");

    for thing in resp.things.unwrap() {
        println!(
            "  Name: {}",
            thing.thing_name.as_deref().unwrap_or_default()
        );
        println!(
            "  Type: {}",
            thing.thing_type_name.as_deref().unwrap_or_default()
        );
        println!(
            "  ARN: {}",
            thing.thing_arn.as_deref().unwrap_or_default()
        );
        println!();
    }
    println!();
    Ok(())
}
```

- For API details, see [ListThings](#) in *AWS SDK for Rust API reference*.

## AWS KMS examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with AWS Key Management Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4437\)](#)

## Actions

### Create a key

The following code example shows how to create an AWS KMS key.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_key(client: &Client) -> Result<(), Error> {
    let resp = client.create_key().send().await?;

    let id = resp
        .key_metadata
        .unwrap()
        .key_id
        .unwrap_or_else(|| String::from("No ID!"));

    println!("Key: {}", id);
    Ok(())
}
```

- For API details, see [CreateKey](#) in *AWS SDK for Rust API reference*.

### Create a random byte string

The following code example shows how to create a random byte string that is cryptographically secure using AWS KMS.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_string(client: &Client, length: i32) -> Result<(), Error> {
    let resp = client
        .generate_random()
```

```
.number_of_bytes(length)
.send()
.await?;

// Did we get an encrypted blob?
let blob = resp.plaintext.expect("Could not get encrypted text");
let bytes = blob.as_ref();

let s = base64::encode(bytes);

println!();
println!("Data key:");
println!("{}", s);

Ok(())
}
```

- For API details, see [GenerateRandom](#) in *AWS SDK for Rust API reference*.

## Decrypt ciphertext

The following code example shows how to decrypt ciphertext that was encrypted by a KMS key.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn decrypt_key(client: &Client, key: &str, filename: &str) -> Result<(), Error> {
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(filename)
        .map(|input| {
            base64::decode(input).expect("Input file does not contain valid base 64
characters.")
        })
        .map(Blob::new);

    let resp = client
        .decrypt()
        .key_id(key)
        .ciphertext_blob(data.unwrap())
        .send()
        .await?;

    let inner = resp.plaintext.unwrap();
    let bytes = inner.as_ref();

    let s = String::from_utf8(bytes.to_vec()).expect("Could not convert to UTF-8");

    println!();
    println!("Decoded string:");
    println!("{}", s);

    Ok(())
}
```

- For API details, see [Decrypt](#) in *AWS SDK for Rust API reference*.

## Encrypt text using a key

The following code example shows how to encrypt text using a KMS key.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn encrypt_string(
    verbose: bool,
    client: &Client,
    text: &str,
    key: &str,
    out_file: &str,
) -> Result<(), Error> {
    let blob = Blob::new(text.as_bytes());

    let resp = client.encrypt().key_id(key).plaintext(blob).send().await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    let mut ofile = File::create(out_file).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");

    if verbose {
        println!("Wrote the following to {:?}", out_file);
        println!("{}", s);
    }
}

Ok(())
}
```

- For API details, see [Encrypt](#) in *AWS SDK for Rust API reference*.

## Generate a plaintext data key for client-side encryption

The following code example shows how to generate a unique symmetric data key for client-side encryption from an AWS KMS key. The key contains both plaintext and ciphertext versions.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);

    Ok(())
}
```

- For API details, see [GenerateDataKey](#) in *AWS SDK for Rust API reference*.

## Generate an encrypted data key

The following code example shows how to generate an encrypted data key from an AWS KMS key. The key contains only a ciphertext version.

### SDK for Rust

**Note**

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_key(client: &Client, key: &str) -> Result<(), Error> {
    let resp = client
        .generate_data_key_without_plaintext()
        .key_id(key)
        .key_spec(DataKeySpec::Aes256)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);

    println!();
    println!("Data key:");
    println!("{}", s);
```

```
        Ok(())
}
```

- For API details, see [GenerateDataKeyWithoutPlaintext](#) in *AWS SDK for Rust API reference*.

## List keys

The following code example shows how to list KMS keys.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_keys(client: &Client) -> Result<(), Error> {
    let resp = client.list_keys().send().await?;

    let keys = resp.keys.unwrap_or_default();

    let len = keys.len();

    for key in keys {
        println!("Key ARN: {}", key.key_arn.as_deref().unwrap_or_default());
    }

    println!();
    println!("Found {} keys", len);

    Ok(())
}
```

- For API details, see [ListKeys](#) in *AWS SDK for Rust API reference*.

## Recrypt ciphertext from one key to another

The following code example shows how to reencrypt ciphertext from one KMS key to another.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn reencrypt_string(
    verbose: bool,
    client: &Client,
```

```
    input_file: &str,
    output_file: &str,
    first_key: &str,
    new_key: &str,
) -> Result<(), Error> {
    // Get blob from input file
    // Open input text file and get contents as a string
    // input is a base-64 encoded string, so decode it:
    let data = fs::read_to_string(input_file)
        .map(|input_file| base64::decode(input_file).expect("invalid base 64"))
        .map(Blob::new);

    let resp = client
        .re_encrypt()
        .ciphertext_blob(data.unwrap())
        .source_key_id(first_key)
        .destination_key_id(new_key)
        .send()
        .await?;

    // Did we get an encrypted blob?
    let blob = resp.ciphertext_blob.expect("Could not get encrypted text");
    let bytes = blob.as_ref();

    let s = base64::encode(bytes);
    let o = &output_file;

    let mut ofile = File::create(o).expect("unable to create file");
    ofile.write_all(s.as_bytes()).expect("unable to write");

    if verbose {
        println!("Wrote the following to {}:", output_file);
        println!("{}", s);
    } else {
        println!("Wrote base64-encoded output to {}", output_file);
    }
}

Ok(())
}
```

- For API details, see [ReEncrypt](#) in *AWS SDK for Rust API reference*.

## Kinesis examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Kinesis.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4442\)](#)

## Actions

### Create a stream

The following code example shows how to create a Kinesis stream.

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_stream(client: &Client, stream: &str) -> Result<(), Error> {
    client
        .create_stream()
        .stream_name(stream)
        .shard_count(4)
        .send()
        .await?;

    println!("Created stream");

    Ok(())
}
```

- For API details, see [CreateStream](#) in *AWS SDK for Rust API reference*.

## Delete a stream

The following code example shows how to delete a Kinesis stream.

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn remove_stream(client: &Client, stream: &str) -> Result<(), Error> {
    client.delete_stream().stream_name(stream).send().await?;

    println!("Deleted stream.");

    Ok(())
}
```

- For API details, see [DeleteStream](#) in *AWS SDK for Rust API reference*.

## Describe a stream

The following code example shows how to describe a Kinesis stream.

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_stream(client: &Client, stream: &str) -> Result<(), Error> {
    let resp = client.describe_stream().stream_name(stream).send().await?;

    let desc = resp.stream_description.unwrap();

    println!("Stream description:");
    println!("  Name:          {}", desc.stream_name.unwrap());
    println!("  Status:        {:?}", desc.stream_status.unwrap());
    println!("  Open shards:   {:?}", desc.shards.unwrap().len());
    println!(
        "  Retention (hours): {}",
        desc.retention_period_hours.unwrap()
    );
    println!("  Encryption:    {:?}", desc.encryption_type.unwrap());

    Ok(())
}
```

- For API details, see [DescribeStream in AWS SDK for Rust API reference](#).

## List streams

The following code example shows how to list information about one or more Kinesis streams.

### SDK for Rust

**Note**

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_streams(client: &Client) -> Result<(), Error> {
    let resp = client.list_streams().send().await?;

    println!("Stream names:");

    let streams = resp.stream_names.unwrap_or_default();
    for stream in &streams {
        println!("  {}", stream);
    }

    println!("Found {} stream(s)", streams.len());

    Ok(())
}
```

- For API details, see [ListStreams in AWS SDK for Rust API reference](#).

## Put data into a stream

The following code example shows how to put data into a Kinesis stream.

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn add_record(client: &Client, stream: &str, key: &str, data: &str) -> Result<(),  
Error> {  
    let blob = Blob::new(data);  
  
    client  
        .put_record()  
        .data(blob)  
        .partition_key(key)  
        .stream_name(stream)  
        .send()  
        .await?;  
  
    println!("Put data into stream.");  
  
    Ok(())
}
```

- For API details, see [PutRecord](#) in *AWS SDK for Rust API reference*.

## MediaLive examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with AWS Elemental MediaLive.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4445\)](#)

## Actions

### List inputs

The following code example shows how to list your MediaLive inputs.

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List your MediaLive input names and ARNs in the Region.

```
async fn show_inputs(client: &Client) -> Result<(), Error> {
    let input_list = client.list_inputs().send().await?;

    for i in input_list.inputs().unwrap_or_default() {
        let input_arn = i.arn().unwrap_or_default();
        let input_name = i.name().unwrap_or_default();

        println!("Input Name : {}", input_name);
        println!("Input ARN : {}", input_arn);
        println!();
    }

    Ok(())
}
```

- For API details, see [ListInputs](#) in *AWS SDK for Rust API reference*.

## MediaPackage examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with AWS Elemental MediaPackage.

**Actions** are code excerpts that show you how to call individual service functions.

**Scenarios** are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4446\)](#)

## Actions

### List channels

The following code example shows how to list your MediaPackage channels.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List channel ARNs and descriptions.

```
async fn show_channels(client: &Client) -> Result<(), Error> {
    let list_channels = client.list_channels().send().await?;

    println!("Channels:");

    for c in list_channels.channels().unwrap_or_default() {
        let description = c.description().unwrap_or_default();
        let arn = c.arn().unwrap_or_default();
```

```
    println!("  Description : {}", description);
    println!("  ARN :          {}", arn);
    println!();
}
Ok(())
}
```

- For API details, see [ListChannels](#) in *AWS SDK for Rust API reference*.

## List your origin endpoints

The following code example shows how to list your MediaPackage origin endpoints.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List your endpoint descriptions and URLs.

```
async fn show_endpoints(client: &Client) -> Result<(), Error> {
    let or_endpoints = client.list_origin_endpoints().send().await?;

    println!("Endpoints:");

    for e in or_endpoints.origin_endpoints().unwrap_or_default() {
        let endpoint_url = e.url().unwrap_or_default();
        let endpoint_description = e.description().unwrap_or_default();
        println!("  Description: {}", endpoint_description);
        println!("  URL :          {}", endpoint_url);
        println!();
    }
    Ok(())
}
```

- For API details, see [ListOriginEndpoints](#) in *AWS SDK for Rust API reference*.

## Amazon Polly examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Polly.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4448\)](#)

## Actions

### Get voices available for synthesis

The following code example shows how to get Amazon Polly voices available for synthesis.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn list_voices(client: &Client) -> Result<(), Error> {
    let resp = client.describe_VOICES().send().await?;

    println!("Voices:");

    let voices = resp.voices().unwrap_or_default();
    for voice in voices {
        println!("  Name: {}",
            voice.name().unwrap_or("No name!"));
        println!(
            "  Language: {}",
            voice.language_name().unwrap_or("No language!"))
    }

    println!();
}

println!("Found {} voices", voices.len());
Ok(())
}
```

- For API details, see [DescribeVoices](#) in *AWS SDK for Rust API reference*.

### List pronunciation lexicons

The following code example shows how to list Amazon Polly pronunciation lexicons.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_lexicons(client: &Client) -> Result<(), Error> {
    let resp = client.list_lexicons().send().await?;

    println!("Lexicons:");

    let lexicons = resp.lexicons().unwrap_or_default();
```

```
for lexicon in lexicons {
    println!("  Name:      {}", lexicon.name().unwrap_or_default());
    println!(
        "  Language: {}?\n",
        lexicon
            .attributes()
            .as_ref()
            .map(|attrib| attrib
                .language_code
                .as_ref()
                .expect("languages must have language codes"))
            .expect("languages must have attributes")
    );
}

println!();
println!("Found {} lexicons.", lexicons.len());
println!();

Ok(())
}
```

- For API details, see [ListLexicons](#) in *AWS SDK for Rust API reference*.

## Store a pronunciation lexicon

The following code example shows how to store an Amazon Polly pronunciation lexicon.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_lexicon(client: &Client, name: &str, from: &str, to: &str) -> Result<(), Error> {
    let content = format!("<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<lexicon version=\"1.0\" xmlns=\"http://www.w3.org/2005/01/pronunciation-lexicon\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xsi:schemaLocation=\"http://www.w3.org/2005/01/pronunciation-lexicon http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd\"
alphabet=\"ipa\" xml:lang=\"en-US\">
<lexeme><grapheme>{}</grapheme><alias>{}</alias></lexeme>
</lexicon>", from, to);

    client
        .put_lexicon()
        .name(name)
        .content(content)
        .send()
        .await?;

    println!("Added lexicon");

    Ok(())
}
```

- For API details, see [PutLexicon](#) in *AWS SDK for Rust API reference*.

## Synthesize speech from text

The following code example shows how to synthesize speech from text with Amazon Polly.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn synthesize(client: &Client, filename: &str) -> Result<(), Error> {
    let content = fs::read_to_string(filename);

    let resp = client
        .synthesize_speech()
        .output_format(OutputFormat::Mp3)
        .text(content.unwrap())
        .voice_id(VoiceId::Joanna)
        .send()
        .await?;

    // Get MP3 data from response and save it
    let mut blob = resp
        .audio_stream
        .collect()
        .await
        .expect("failed to read data");

    let parts: Vec<&str> = filename.split('.').collect();
    let out_file = format!("{}{}", String::from(parts[0]), ".mp3");

    let mut file = tokio::fs::File::create(out_file)
        .await
        .expect("failed to create file");

    file.write_all_buf(&mut blob)
        .await
        .expect("failed to write to file");

    Ok(())
}
```

- For API details, see [SynthesizeSpeech](#) in *AWS SDK for Rust API reference*.

## QLDB examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon QLDB.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4451\)](#)

## Actions

### Create a ledger

The following code example shows how to create a QLDB ledger.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_ledger(client: &Client, ledger: &str) -> Result<(), Error> {
    let result = client
        .create_ledger()
        .name(ledger)
        .permissions_mode(PermissionsMode::AllowAll)
        .send()
        .await?;

    println!("ARN: {}", result.arn().unwrap());

    Ok(())
}
```

- For API details, see [CreateLedger](#) in *AWS SDK for Rust API reference*.

### List your ledgers

The following code example shows how to list your QLDB ledgers.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_ledgers(client: &QLDBClient) -> Result<(), Error> {
    let mut pages = client.list_ledgers().intoPaginator().page_size(2).send();

    while let Some(page) = pages.next().await {
        println!("* {:?}", page); //Prints an entire page of ledgers.
        for ledger in page.unwrap().ledgers().unwrap() {
```

```
        println!("* {:?}", ledger); //Prints the LedgerSummary of a single ledger.
    }
}

Ok(())
}
```

- For API details, see [ListLedgers](#) in *AWS SDK for Rust API reference*.

## Amazon RDS Data Service examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Relational Database Service Data Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4452\)](#)

## Actions

### Run an SQL statement

The following code example shows how to run an SQL statement.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn query_cluster(
    client: &Client,
    cluster_arn: &str,
    query: &str,
    secret_arn: &str,
) -> Result<(), Error> {
    let st = client
        .execute_statement()
        .resource_arn(cluster_arn)
        .database("postgres") // Do not confuse this with db instance name
        .sql(query)
        .secret_arn(secret_arn);

    let result = st.send().await?;

    println!("{:?}", result);
    println!();

    Ok(())
}
```

```
}
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Rust API reference*.

## Route 53 examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Route 53.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4453\)](#)

## Actions

### Get a list of the public and private hosted zones

The following code example shows how to get a list of the public and private hosted zones.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_host_info(client: &aws_sdk_route53::Client) -> Result<(),  
aws_sdk_route53::Error> {  
    let hosted_zone_count = client.get_hosted_zone_count().send().await?;  
  
    println!(  
        "Number of hosted zones in region : {}",  
        hosted_zone_count.hosted_zone_count().unwrap_or_default(),  
    );  
  
    let hosted_zones = client.list_hosted_zones().send().await?;  
  
    println!("Zones:");  
  
    for hz in hosted_zones.hosted_zones().unwrap_or_default() {  
        let zone_name = hz.name().unwrap_or_default();  
        let zone_id = hz.id().unwrap_or_default();  
  
        println!("  ID :  {}", zone_id);  
        println!("  Name : {}", zone_name);  
        println!();  
    }  
  
    Ok(())
}
```

- For API details, see [ListHostedZones](#) in *AWS SDK for Rust API reference*.

## Amazon S3 examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Simple Storage Service.

**Actions** are code excerpts that show you how to call individual service functions.

**Scenarios** are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4454\)](#)
- [Scenarios \(p. 4462\)](#)

## Actions

### Complete a multipart upload

The following code example shows how to complete a multipart upload action.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await
    .unwrap();
```

- For API details, see [CompleteMultipartUpload](#) in *AWS SDK for Rust API reference*.

### Copy an object from one bucket to another

The following code example shows how to copy an S3 object from one bucket to another.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn copy_object(
    client: &Client,
    bucket_name: &str,
    object_key: &str,
    target_key: &str,
) -> Result<(), Error> {
    let mut source_bucket_and_object: String = "".to_owned();
    source_bucket_and_object.push_str(bucket_name);
    source_bucket_and_object.push('/');
    source_bucket_and_object.push_str(object_key);

    client
        .copy_object()
        .copy_source(source_bucket_and_object)
        .bucket(bucket_name)
        .key(target_key)
        .send()
        .await?;

    Ok(())
}
```

- For API details, see [CopyObject](#) in *AWS SDK for Rust API reference*.

## Create a bucket

The following code example shows how to create an S3 bucket.

### SDK for Rust

**Note**

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn create_bucket(client: &Client, bucket_name: &str, region: &str) ->
Result<(), Error> {
    let constraint = BucketLocationConstraint::from(region);
    let cfg = CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await?;
    println!("Creating bucket named: {}", bucket_name);
    Ok(())
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for Rust API reference*.

## Create a multipart upload

The following code example shows how to create the structure to build a multipart upload action.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await
    .unwrap();
```

- For API details, see [CreateMultipartUpload](#) in *AWS SDK for Rust API reference*.

## Delete an empty bucket

The following code example shows how to delete an empty S3 bucket.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_bucket(client: &Client, bucket_name: &str) -> Result<(), Error> {
    client.delete_bucket().bucket(bucket_name).send().await?;
    println!("Bucket deleted");
    Ok(())
}
```

- For API details, see [DeleteBucket](#) in *AWS SDK for Rust API reference*.

## Delete an object

The following code example shows how to delete an S3 object.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn remove_object(client: &Client, bucket: &str, key: &str) -> Result<(), Error> {
    client
        .delete_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await?;

    println!("Object deleted.");
    Ok(())
}
```

- For API details, see [DeleteObject](#) in *AWS SDK for Rust API reference*.

## Delete multiple objects

The following code example shows how to delete multiple objects from an S3 bucket.

### SDK for Rust

**Note**

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn delete_objects(client: &Client, bucket_name: &str) -> Result<(), Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    let mut delete_objects: Vec<ObjectIdentifier> = vec![];
    for obj in objects.contents().unwrap_or_default() {
        let obj_id = ObjectIdentifier::builder()
            .set_key(Some(obj.key().unwrap().to_string()))
            .build();
        delete_objects.push(obj_id);
    }
    client
        .delete_objects()
        .bucket(bucket_name)
        .delete(Delete::builder().set_objects(Some(delete_objects)).build())
        .send()
        .await?;

    let objects: ListObjectsV2Output =
        client.list_objects_v2().bucket(bucket_name).send().await?;
    match objects.key_count {
        0 => Ok(()),
        _ => Err(Error::Unhandled(Box::from(
            "There were still objects left in the bucket.",
        ))),
    }
}
```

- For API details, see [DeleteObjects](#) in *AWS SDK for Rust API reference*.

## Get an object from a bucket

The following code example shows how to read data from an object in an S3 bucket.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn download_object(client: &Client, bucket_name: &str, key: &str) ->
    GetObjectOutput {
    let resp = client
        .get_object()
        .bucket(bucket_name)
        .key(key)
        .send()
        .await;
    resp.unwrap()
}
```

- For API details, see [GetObject](#) in *AWS SDK for Rust API reference*.

## Get the Region location for a bucket

The following code example shows how to get the Region location for an S3 bucket.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_buckets(strict: bool, client: &Client, region: &str) -> Result<(), Error> {
    let resp = client.list_buckets().send().await?;
    let buckets = resp.buckets().unwrap_or_default();
    let num_buckets = buckets.len();

    let mut in_region = 0;

    for bucket in buckets {
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
```

```
.await?;

    if r.location_constraint().unwrap().as_ref() == region {
        println!("{}", bucket.name().unwrap_or_default());
        in_region += 1;
    }
} else {
    println!("{}", bucket.name().unwrap_or_default());
}
}

println!();
if strict {
    println!(
        "Found {} buckets in the {} region out of a total of {} buckets.",
        in_region, region, num_buckets
    );
} else {
    println!("Found {} buckets in all regions.", num_buckets);
}

Ok(())
}
```

- For API details, see [GetBucketLocation](#) in *AWS SDK for Rust API reference*.

## List buckets

The following code example shows how to list S3 buckets.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_buckets(strict: bool, client: &Client, region: &str) -> Result<(), Error> {
    let resp = client.list_buckets().send().await?;
    let buckets = resp.buckets().unwrap_or_default();
    let num_buckets = buckets.len();

    let mut in_region = 0;

    for bucket in buckets {
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
                .await?;

            if r.location_constraint().unwrap().as_ref() == region {
                println!("{}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{}", bucket.name().unwrap_or_default());
        }
    }
}
```

```
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            num_buckets, in_region, region
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}
```

- For API details, see [ListBuckets](#) in *AWS SDK for Rust API reference*.

## List object versions in a bucket

The following code example shows how to list object versions in an S3 bucket.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {
    let resp = client.list_object_versions().bucket(bucket).send().await?;

    for version in resp.versions().unwrap_or_default() {
        println!("{}:", version.key().unwrap_or_default());
        println!("  version ID: {}", version.version_id().unwrap_or_default());
        println!();
    }

    Ok(())
}
```

- For API details, see [ListObjectVersions](#) in *AWS SDK for Rust API reference*.

## List objects in a bucket

The following code example shows how to list objects in an S3 bucket.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn list_objects(client: &Client, bucket_name: &str) -> Result<(), Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;
    println!("Objects in bucket:");
    for obj in objects.contents().unwrap_or_default() {
        println!("{}: {}", obj.key().unwrap());
    }
}
Ok(())
}
```

- For API details, see [ListObjects](#) in *AWS SDK for Rust API reference*.

## Upload a single part of a multipart upload

The following code example shows how to upload a single part of a multipart upload.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
let upload_part_res = client
    .upload_part()
    .key(&key)
    .bucket(&bucket_name)
    .upload_id(upload_id)
    .body(stream)
    .part_number(part_number)
    .send()
    .await?;
upload_parts.push(
    CompletedPart::builder()
        .e_tag(upload_part_res.e_tag.unwrap_or_default())
        .part_number(part_number)
        .build(),
);
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();
```

- For API details, see [UploadPart](#) in *AWS SDK for Rust API reference*.

## Upload an object to a bucket

The following code example shows how to upload an object to an S3 bucket.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
pub async fn upload_object(
    client: &Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<(), Error> {
    let body = ByteStream::from_path(Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await?;

    println!("Uploaded file: {}", file_name);
    Ok(())
}
```

- For API details, see [PutObject](#) in *AWS SDK for Rust API reference*.

## Scenarios

### Get started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.
- Delete a bucket.

### SDK for Rust

**Note**

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Code for the binary crate which runs the scenario.

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::{Client, Error, Region};
```

```

use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), Error> {
    let (region, client, bucket_name, file_name, key, target_key) =
initialize_variables().await;

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name, key,
target_key).await
    {
        println!("{}:{}", e);
    };

    Ok(())
}

async fn initialize_variables() -> (Region, Client, String, String, String, String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-west-2"));
    let region = region_provider.region().await.unwrap();

    let shared_config = aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);

    let bucket_name = format!("doc-example-bucket-{}", Uuid::new_v4());

    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
    let target_key = "target_key".to_string();

    (region, client, bucket_name, file_name, key, target_key)
}

async fn run_s3_operations(
    region: Region,
    client: Client,
    bucket_name: String,
    file_name: String,
    key: String,
    target_key: String,
) -> Result<(), Error> {
    s3_service::create_bucket(&client, &bucket_name, region.as_ref()).await?;
    s3_service::upload_object(&client, &bucket_name, &file_name, &key).await?;
    let _object = s3_service::download_object(&client, &bucket_name, &key).await?;
    s3_service::copy_object(&client, &bucket_name, &key, &target_key).await?;
    s3_service::list_objects(&client, &bucket_name).await?;
    s3_service::delete_objects(&client, &bucket_name).await?;
    s3_service::delete_bucket(&client, &bucket_name).await?;

    Ok(())
}

```

A library crate with common actions called by the binary.

```

use aws_sdk_s3::model::{
    BucketLocationConstraint, CreateBucketConfiguration, Delete, ObjectIdentifier,
};
use aws_sdk_s3::output::{GetObjectOutput, ListObjectsV2Output};
use aws_sdk_s3::types::ByteStream;
use aws_sdk_s3::{Client, Error};
use std::path::Path;
use std::str;

```

```
pub async fn delete_bucket(client: &Client, bucket_name: &str) -> Result<(), Error> {
    client.delete_bucket().bucket(bucket_name).send().await?;
    println!("Bucket deleted");
    Ok(())
}

pub async fn delete_objects(client: &Client, bucket_name: &str) -> Result<(), Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    let mut delete_objects: Vec<ObjectIdentifier> = vec![];
    for obj in objects.contents().unwrap_or_default() {
        let obj_id = ObjectIdentifier::builder()
            .set_key(Some(obj.key().unwrap().to_string()))
            .build();
        delete_objects.push(obj_id);
    }
    client
        .delete_objects()
        .bucket(bucket_name)
        .delete(Delete::builder().set_objects(Some(delete_objects)).build())
        .send()
        .await?;

    let objects: ListObjectsV2Output =
        client.list_objects_v2().bucket(bucket_name).send().await?;
    match objects.key_count {
        0 => Ok(()),
        _ => Err(Error::Unhandled(Box::from(
            "There were still objects left in the bucket.",
        ))),
    }
}

pub async fn list_objects(client: &Client, bucket_name: &str) -> Result<(), Error> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;
    println!("Objects in bucket:");
    for obj in objects.contents().unwrap_or_default() {
        println!("{}:", obj.key().unwrap());
    }

    Ok(())
}

pub async fn copy_object(
    client: &Client,
    bucket_name: &str,
    object_key: &str,
    target_key: &str,
) -> Result<(), Error> {
    let mut source_bucket_and_object: String = "".to_owned();
    source_bucket_and_object.push_str(bucket_name);
    source_bucket_and_object.push('/');
    source_bucket_and_object.push_str(object_key);

    client
        .copy_object()
        .copy_source(source_bucket_and_object)
        .bucket(bucket_name)
        .key(target_key)
        .send()
        .await?;

    Ok(())
}
```

```
pub async fn download_object(client: &Client, bucket_name: &str, key: &str) ->
    GetObjectOutput {
    let resp = client
        .get_object()
        .bucket(bucket_name)
        .key(key)
        .send()
        .await;
    resp.unwrap()
}

pub async fn upload_object(
    client: &Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<(), Error> {
    let body = ByteStream::from_path(Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await?;

    println!("Uploaded file: {}", file_name);
    Ok(())
}

pub async fn create_bucket(client: &Client, bucket_name: &str, region: &str) ->
    Result<(), Error> {
    let constraint = BucketLocationConstraint::from(region);
    let cfg = CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await?;
    println!("Creating bucket named: {}", bucket_name);
    Ok(())
}
```

- For API details, see the following topics in *AWS SDK for Rust API reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)

## Upload or download large files

The following code example shows how to upload or download large files to and from Amazon S3.

For more information, see [Uploading an object using multipart upload](#).

## SDK for Rust

### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
use std::convert::TryInto;
use std::fs::File;
use std::io::prelude::*;
use std::path::Path;

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::model::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::output::{CreateMultipartUploadOutput, GetObjectOutput};
use aws_sdk_s3::{Client as S3Client, Error, Region};
use aws_smithy_http::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::{thread_rng, Rng};
use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() -> Result<(), Error> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("doc-example-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-west-2"));
    let region = region_provider.region().await.unwrap();
    s3_service::create_bucket(&client, &bucket_name, region.as_ref()).await?;

    let key = "sample.txt".to_string();
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await
        .unwrap();
    let upload_id = multipart_upload_res.upload_id().unwrap();

    //Create a file of random characters for the upload.
    let mut file = File::create(&key).expect("Could not create sample file.");
    // Loop until the file is 5 chunks.
    while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
        let rand_string: String = thread_rng()
            .sample_iter(&Alphanumeric)
            .take(256)
            .map(char::from)
            .collect();
        let return_string: String = "\n".to_string();
        file.write_all(rand_string.as_ref())
            .expect("Error writing to file.");
        file.write_all(return_string.as_ref())
            .expect("Error writing to file.");
    }
}
```

```
let path = Path::new(&key);
let file_size = tokio::fs::metadata(path)
    .await
    .expect("it exists I swear")
    .len();

let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
let mut size_of_last_chunk = file_size % CHUNK_SIZE;
if size_of_last_chunk == 0 {
    size_of_last_chunk = CHUNK_SIZE;
    chunk_count -= 1;
}

if file_size == 0 {
    panic!("Bad file size.");
}
if chunk_count > MAX_CHUNKS {
    panic!("Too many chunks! Try increasing your chunk size.")
}

let mut upload_parts: Vec<CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();
    //Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
        .await?;
    upload_parts.push(
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
            .part_number(part_number)
            .build(),
    );
}
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
```

```
.await
.unwrap();

let data: GetObjectOutput = s3_service::download_object(&client, &bucket_name,
&key).await;
let data_length: u64 = data.content_length().try_into().unwrap();
if file.metadata().unwrap().len() == data_length {
    println!("Data lengths match.");
} else {
    println!("The data was not the same size!");
}

s3_service::delete_objects(&client, &bucket_name)
.await
.expect("Error emptying bucket.");
s3_service::delete_bucket(&client, &bucket_name)
.await
.expect("Error deleting bucket.");

Ok(())
}
```

## Amazon SES API v2 examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Simple Email Service API v2.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4468\)](#)

## Actions

### Create a contact in a contact list

The following code example shows how to create an Amazon SES API v2 contact in a contact list.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn add_contact(client: &Client, list: &str, email: &str) -> Result<(), Error> {
    client
        .create_contact()
        .contact_list_name(list)
        .email_address(email)
        .send()
        .await?;
```

```
    println!("Created contact");

    Ok(())
}
```

- For API details, see [CreateContact](#) in *AWS SDK for Rust API reference*.

## Create a contact list

The following code example shows how to create an Amazon SES API v2 contact list.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_list(client: &Client, contact_list: &str) -> Result<(), Error> {
    client
        .create_contact_list()
        .contact_list_name(contact_list)
        .send()
        .await?;

    println!("Created contact list.");

    Ok(())
}
```

- For API details, see [CreateContactList](#) in *AWS SDK for Rust API reference*.

## Get identity information

The following code example shows how to get Amazon SES API v2 identity information.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Determines whether an email address has been verified.

```
async fn is_verified(client: &Client, email: &str) -> Result<(), Error> {
    let resp = client
        .get_email_identity()
        .email_identity(email)
        .send()
```

```
.await?;

if resp.verified_for_sending_status() {
    println!("The address is verified");
} else {
    println!("The address is not verified");
}

Ok(())
}
```

- For API details, see [GetEmailIdentity](#) in *AWS SDK for Rust API reference*.

## List the contact lists

The following code example shows how to list the Amazon SES API v2 contact lists.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_lists(client: &Client) -> Result<(), Error> {
    let resp = client.list_contact_lists().send().await?;

    println!("Contact lists:");

    for list in resp.contact_lists().unwrap_or_default() {
        println!("  {}", list.contact_list_name().unwrap_or_default());
    }

    Ok(())
}
```

- For API details, see [ListContactLists](#) in *AWS SDK for Rust API reference*.

## List the contacts in a contact list

The following code example shows how to list the contacts in an Amazon SES API v2 contact list.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_contacts(client: &Client, list: &str) -> Result<(), Error> {
```

```
let resp = client
    .list_contacts()
    .contact_list_name(list)
    .send()
    .await?;

println!("Contacts:");

for contact in resp.contacts().unwrap_or_default() {
    println!("  {}", contact.email_address().unwrap_or_default());
}

Ok(())
}
```

- For API details, see [ListContacts](#) in *AWS SDK for Rust API reference*.

## Send an email

The following code example shows how to send an Amazon SES API v2 email.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Sends a message to all members of the contact list.

```
async fn send_message(
    client: &Client,
    list: &str,
    from: &str,
    subject: &str,
    message: &str,
) -> Result<(), Error> {
    // Get list of email addresses from contact list.
    let resp = client
        .list_contacts()
        .contact_list_name(list)
        .send()
        .await?;

    let contacts = resp.contacts().unwrap_or_default();

    let cs: String = contacts
        .iter()
        .map(|i| i.email_address().unwrap_or_default())
        .collect();

    let dest = Destination::builder().to_addresses(cs).build();
    let subject_content = Content::builder().data(subject).charset("UTF-8").build();
    let body_content = Content::builder().data(message).charset("UTF-8").build();
    let body = Body::builder().text(body_content).build();

    let msg = Message::builder()
        .subject(subject_content)
        .body(body)
```

```
.build();

let email_content = EmailContent::builder().simple(msg).build();

client
    .send_email()
    .from_email_address(from)
    .destination(dest)
    .content(email_content)
    .send()
    .await?;

println!("Email sent to list");

Ok(())
}
```

- For API details, see [SendEmail](#) in *AWS SDK for Rust API reference*.

## Amazon SNS examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Simple Notification Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4472\)](#)

## Actions

### Create a topic

The following code example shows how to create an Amazon SNS topic.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );
}

Ok(())
}
```

- For API details, see [CreateTopic](#) in *AWS SDK for Rust API reference*.

## List topics

The following code example shows how to list Amazon SNS topics.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_topics(client: &Client) -> Result<(), Error> {
    let resp = client.list_topics().send().await?;

    println!("Topic ARNs:");

    for topic in resp.topics().unwrap_or_default() {
        println!("{}", topic.topic_arn().unwrap_or_default());
    }

    Ok(())
}
```

- For API details, see [ListTopics](#) in *AWS SDK for Rust API reference*.

## Publish to a topic

The following code example shows how to publish messages to an Amazon SNS topic.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}'", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
```

```
.await?;

println!("Added a subscription: {:?}", rsp);

let rsp = client
.publish()
.topic_arn(topic_arn)
.message("hello sns!")
.send()
.await?;

println!("Published message: {:?}", rsp);

Ok(())
}
```

- For API details, see [Publish](#) in *AWS SDK for Rust API reference*.

## Subscribe an email address to a topic

The following code example shows how to subscribe an email address to an Amazon SNS topic.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`, topic_arn");

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- For API details, see [Subscribe](#) in *AWS SDK for Rust API reference*.

## Amazon SQS examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon Simple Queue Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4475\)](#)

## Actions

### List queues

The following code example shows how to list Amazon SQS queues.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Retrieve the first Amazon SQS queue listed in the Region.

```
async fn find_first_queue(client: &Client) -> Result<String, Error> {
    let queues = client.list_queues().send().await?;
    let queue_urls = queues.queue_urls().unwrap_or_default();
    Ok(queue_urls
        .first()
        .expect("No queues in this account and Region. Create a queue to proceed.")
        .to_string())
}
```

- For API details, see [ListQueues](#) in *AWS SDK for Rust API reference*.

### Receive messages from a queue

The following code example shows how to receive messages from an Amazon SQS queue.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn receive(client: &Client, queue_url: &String) -> Result<(), Error> {
    let rcv_message_output =
        client.receive_message().queue_url(queue_url).send().await?;

    println!("Messages from queue with url: {}", queue_url);

    for message in rcv_message_output.messages.unwrap_or_default() {
        println!("Got the message: {:?}", message);
    }

    Ok(())
}
```

- For API details, see [ReceiveMessage in AWS SDK for Rust API reference](#).

## Send a message to a queue

The following code example shows how to send a message to an Amazon SQS queue.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn send(client: &Client, queue_url: &String, message: &SQSMessage) -> Result<(), Error> {
    println!("Sending message to queue with URL: {}", queue_url);

    let rsp = client
        .send_message()
        .queue_url(queue_url)
        .message_body(&message.body)
        .message_group_id(&message.group)
        // If the queue is FIFO, you need to set .message_deduplication_id
        // or configure the queue for ContentBasedDeduplication.
        .send()
        .await?;

    println!("Send message to the queue: {:?}", rsp);

    Ok(())
}
```

- For API details, see [SendMessage in AWS SDK for Rust API reference](#).

## SageMaker examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with Amazon SageMaker.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4477\)](#)

## Actions

### List notebook instances

The following code example shows how to list SageMaker notebook instances.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_instances(client: &Client) -> Result<(), Error> {
    let notebooks = client.list_notebook_instances().send().await?;

    println!("Notebooks:");

    for n in notebooks.notebook_instances().unwrap_or_default() {
        let n_instance_type = n.instance_type().unwrap();
        let n_status = n.notebook_instance_status().unwrap();
        let n_name = n.notebook_instance_name().unwrap_or_default();

        println!("  Name : {}", n_name);
        println!("  Status : {}", n_status.as_ref());
        println!("  Instance Type : {}", n_instance_type.as_ref());
        println!();
    }

    Ok(())
}
```

- For API details, see [ListNotebookInstances](#) in *AWS SDK for Rust API reference*.

### List training jobs

The following code example shows how to list SageMaker training jobs.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_jobs(client: &Client) -> Result<(), Error> {
```

```
let job_details = client.list_training_jobs().send().await?;

println!("Jobs:");

for j in job_details.training_job_summaries().unwrap_or_default() {
    let name = j.training_job_name().unwrap_or_default();
    let creation_time = j.creation_time().unwrap().to_chrono_utc();
    let training_end_time = j.training_end_time().unwrap().to_chrono_utc();

    let status = j.training_job_status().unwrap();
    let duration = training_end_time - creation_time;

    println!("  Name:          {}", name);
    println!("    Creation date/time: {}",
            creation_time.format("%Y-%m-%d@%H:%M:%S")
        );
    println!("  Duration (seconds): {}", duration.num_seconds());
    println!("  Status:         {}", status.as_ref());

    println!();
}

Ok(())
}
```

- For API details, see [ListTrainingJobs](#) in *AWS SDK for Rust API reference*.

## Secrets Manager examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with AWS Secrets Manager.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4478\)](#)

## Actions

### Create a secret

The following code example shows how to create a Secrets Manager secret.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_secret(client: &Client, name: &str, value: &str) -> Result<(), Error> {
```

```
client
    .create_secret()
    .name(name)
    .secret_string(value)
    .send()
    .await?;

    println!("Created secret");

    Ok(())
}
```

- For API details, see [CreateSecret](#) in *AWS SDK for Rust API reference*.

## Get a secret value

The following code example shows how to get a Secrets Manager secret value.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_secret(client: &Client, name: &str) -> Result<(), Error> {
    let resp = client.get_secret_value().secret_id(name).send().await?;

    println!("Value: {}", resp.secret_string().unwrap_or("No value!"));

    Ok(())
}
```

- For API details, see [GetSecretValue](#) in *AWS SDK for Rust API reference*.

## List secrets

The following code example shows how to list Secrets Manager secrets.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_secrets(client: &Client) -> Result<(), Error> {
    let resp = client.list_secrets().send().await?;

    println!("Secret names:");
}
```

```
let secrets = resp.secret_list().unwrap_or_default();
for secret in secrets {
    println!("  {}", secret.name().unwrap_or("No name!"));
}

println!("Found {} secrets", secrets.len());
Ok(())
}
```

- For API details, see [ListSecrets](#) in *AWS SDK for Rust API reference*.

## Systems Manager examples using SDK for Rust

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Rust with AWS Systems Manager.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4480\)](#)

## Actions

### Add a parameter

The following code example shows how to add a Systems Manager parameter.

#### SDK for Rust

##### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn make_parameter(
    client: &Client,
    name: &str,
    value: &str,
    description: &str,
) -> Result<(), Error> {
    let resp = client
        .put_parameter()
        .overwrite(true)
        .r#type(ParameterType::String)
        .name(name)
        .value(value)
        .description(description)
        .send()
        .await?;
```

```
    println!("Success! Parameter now has version: {}", resp.version());  
    Ok(())  
}
```

- For API details, see [PutParameter](#) in *AWS SDK for Rust API reference*.

## Get parameters information

The following code example shows how to get Systems Manager parameters information.

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
async fn show_parameters(client: &Client) -> Result<(), Error> {  
    let resp = client.describe_parameters().send().await?;  
  
    for param in resp.parameters().unwrap().iter() {  
        println!("  {}", param.name().unwrap_or_default());  
    }  
  
    Ok(())  
}
```

- For API details, see [DescribeParameters](#) in *AWS SDK for Rust API reference*.

## Cross-service examples using SDK for Rust

The following sample applications use the AWS SDK for Rust to work across multiple AWS services.

### Examples

- [Convert text to speech and back to text using an AWS SDK \(p. 4481\)](#)
- [Detect faces in an image using an AWS SDK \(p. 4482\)](#)
- [Save EXIF and other image information using an AWS SDK \(p. 4482\)](#)

## Convert text to speech and back to text using an AWS SDK

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

Use Amazon Polly to synthesize a plain text (UTF-8) input file to an audio file, upload the audio file to an Amazon Simple Storage Service bucket, use Amazon Transcribe to convert that audio file to text, and display the text.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Polly
- Amazon S3
- Amazon Transcribe

## Detect faces in an image using an AWS SDK

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

Save the image in an Amazon Simple Storage Service bucket with an **uploads** prefix, use Amazon Rekognition to detect facial details, such as age range, gender, and emotion (smiling, etc.), and display those details.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- Amazon Rekognition
- Amazon S3

## Save EXIF and other image information using an AWS SDK

### SDK for Rust

#### Note

This documentation is for an SDK in preview release. The SDK is subject to change and should not be used in production.

Get EXIF information from a JPG, JPEG, or PNG file, upload the image file to an Amazon Simple Storage Service bucket, use Amazon Rekognition to identify the three top attributes (*labels* in Amazon Rekognition) in the file, and add the EXIF and label information to a Amazon DynamoDB table in the Region.

For complete source code and instructions on how to set up and run, see the full example on [GitHub](#).

### Services used in this example

- DynamoDB
- Amazon Rekognition
- Amazon S3

## Code examples for SDK for SAP ABAP

The code examples in this topic show you how to use the AWS SDK for SAP ABAP with AWS.

### Examples

- [Single-service actions and scenarios using SDK for SAP ABAP \(p. 4483\)](#)

# Single-service actions and scenarios using SDK for SAP ABAP

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with AWS services.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Services

- [CloudWatch examples using SDK for SAP ABAP \(p. 4483\)](#)
- [Amazon EC2 examples using SDK for SAP ABAP \(p. 4488\)](#)
- [Kinesis examples using SDK for SAP ABAP \(p. 4499\)](#)
- [Lambda examples using SDK for SAP ABAP \(p. 4506\)](#)
- [Amazon S3 examples using SDK for SAP ABAP \(p. 4516\)](#)
- [Amazon SNS examples using SDK for SAP ABAP \(p. 4522\)](#)
- [Amazon SQS examples using SDK for SAP ABAP \(p. 4528\)](#)
- [SageMaker examples using SDK for SAP ABAP \(p. 4531\)](#)
- [Amazon Textract examples using SDK for SAP ABAP \(p. 4544\)](#)
- [Amazon Translate examples using SDK for SAP ABAP \(p. 4551\)](#)

## CloudWatch examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon CloudWatch.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Topics

- [Actions \(p. 4483\)](#)
- [Scenarios \(p. 4486\)](#)

## Actions

### Create an alarm that watches a metric

The following code example shows how to create an Amazon CloudWatch alarm that watches a metric.

#### SDK for SAP ABAP

##### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_cwt->putmetricalarm(  
        iv_alarmname          = iv_alarm_name  
        iv_comparisonoperator = iv_comparison_operator  
        iv_evaluationperiods = iv_evaluation_periods  
        iv_metricname         = iv_metric_name  
        iv_namespace          = iv_namespace  
        iv_statistic          = iv_statistic  
        iv_threshold          = iv_threshold  
        iv_actionsenabled     = iv_actions_enabled  
        iv_alarmdescription   = iv_alarm_description  
        iv_unit                = iv_unit  
        iv_period              = iv_period  
        it_dimensions          = it_dimensions  
    ).  
    MESSAGE 'Alarm created' TYPE 'I'.  
    CATCH /aws1/cx_cwtlimitexceededfault.  
        MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.  
    ENDTRY.
```

- For API details, see [PutMetricAlarm](#) in *AWS SDK for SAP ABAP API reference*.

## Delete alarms

The following code example shows how to delete Amazon CloudWatch alarms.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_cwt->deletealarms(  
        it_alarmnames = it_alarm_names  
    ).  
    MESSAGE 'Alarms deleted' TYPE 'I'.  
    CATCH /aws1/cx_cwtresourcenotfound .  
        MESSAGE 'Resource being access is not found.' TYPE 'E'.  
    ENDTRY.
```

- For API details, see [DeleteAlarms](#) in *AWS SDK for SAP ABAP API reference*.

## Describe alarms

The following code example shows how to describe Amazon CloudWatch alarms.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_cwt->describealarms(                                " oo_result is returned for  
testing purpose "  
        it_alarmnames = it_alarm_names  
    ).  
    MESSAGE 'Alarms retrieved' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.  
        MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [DescribeAlarms in AWS SDK for SAP ABAP API reference](#).

## Disable alarm actions

The following code example shows how to disable Amazon CloudWatch alarm actions.

### SDK for SAP ABAP

**Note**

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Disables actions on the specified alarm. "  
TRY.  
    lo_cwt->disablealarmactions(  
        it_alarmnames = it_alarm_names  
    ).  
    MESSAGE 'Alarm actions disabled' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.  
        MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [DisableAlarmActions in AWS SDK for SAP ABAP API reference](#).

## Enable alarm actions

The following code example shows how to enable Amazon CloudWatch alarm actions.

### SDK for SAP ABAP

**Note**

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Enable actions on the specified alarm."  
TRY.  
    lo_cwt->enablealarmactions(  
        it_alarmnames = it_alarm_names  
    ).  
    MESSAGE 'Alarm actions enabled' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.  
        MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [EnableAlarmActions](#) in *AWS SDK for SAP ABAP API reference*.

## List metrics

The following code example shows how to list Amazon CloudWatch metrics.

### SDK for SAP ABAP

**Note**

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"The following list-metrics example displays the metrics for Amazon Cloudwatch."  
TRY.  
    oo_result = lo_cwt->listmetrics(                                     " oo_result is returned for testing  
purpose "  
        iv_namespace = iv_namespace  
    ).  
    DATA(lt_metrics) = oo_result->get_metrics( ).  
    MESSAGE 'Metrics retrieved' TYPE 'I'.  
    CATCH /aws1/cx_cwtinvparamvalueex .  
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.  
    ENDTRY.
```

- For API details, see [ListMetrics](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios

### Get started with alarms

The following code example shows how to:

- Create an alarm.
- Disable alarm actions.
- Describe an alarm.

- Delete an alarm.

## SDK for SAP ABAP

### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lt_alarmnames TYPE /aws1/cl_cwtalarmnames_w=>tt_alarmnames.  
DATA lo_alarmname TYPE REF TO /aws1/cl_cwtalarmnames_w.  
  
"Create an alarm"  
TRY.  
    lo_cwt->putmetricalarm(  
        iv_alarmname          = iv_alarm_name  
        iv_comparisonoperator = iv_comparison_operator  
        iv_evaluationperiods = iv_evaluation_periods  
        iv_metricname         = iv_metric_name  
        iv_namespace          = iv_namespace  
        iv_statistic          = iv_statistic  
        iv_threshold          = iv_threshold  
        iv_actionsenabled     = iv_actions_enabled  
        iv_alarmdescription   = iv_alarm_description  
        iv_unit               = iv_unit  
        iv_period              = iv_period  
        it_dimensions          = it_dimensions  
    ).  
    MESSAGE 'Alarm created' TYPE 'I'.  
    CATCH /aws1/cx_cwtlimitexceededfault.  
        MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.  
    ENDTRY.  
  
"Create an ABAP internal table for the alarm created"  
CREATE OBJECT lo_alarmname EXPORTING iv_value = iv_alarm_name.  
INSERT lo_alarmname INTO TABLE lt_alarmnames.  
  
"Disable alarm actions"  
TRY.  
    lo_cwt->disablealarmactions(  
        it_alarmnames          = lt_alarmnames  
    ).  
    MESSAGE 'Alarm actions disabled' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_disablealarm_exception).  
        DATA(lv_disablealarm_error) = |"{" lo_disablealarm_exception->av_err_code }" -  
        { lo_disablealarm_exception->av_err_msg }|.  
        MESSAGE lv_disablealarm_error TYPE 'E'.  
    ENDTRY.  
  
"Describe alarm using the same ABAP internal table"  
TRY.  
    oo_result = lo_cwt->describealarms(                                     " oo_result is  
    returned for testing purpose "  
        it_alarmnames          = lt_alarmnames  
    ).  
    MESSAGE 'Alarms retrieved' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_describealarms_exception).  
        DATA(lv_describealarms_error) = |"{" lo_describealarms_exception->av_err_code }" -  
        { lo_describealarms_exception->av_err_msg }|.
```

```
MESSAGE lv_describealarms_error TYPE 'E'.
ENDTRY.

"Delete alarm"
TRY.
    lo_cwt->deletealarms(
        it_alarmnames = lt_alarmnames
    ).
    MESSAGE 'Alarms deleted' TYPE 'I'.
    CATCH /aws1/cx_cwtresourcenotfound .
    MESSAGE 'Resource being access is not found.' TYPE 'E'.
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [DeleteAlarms](#)
  - [DescribeAlarms](#)
  - [DisableAlarmActions](#)
  - [PutMetricAlarm](#)

## Amazon EC2 examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon Elastic Compute Cloud.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4488\)](#)

## Actions

### Allocate an Elastic IP address

The following code example shows how to allocate an Elastic IP address for Amazon EC2.

#### SDK for SAP ABAP

##### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_ec2->allocateaddress( iv_domain = 'vpc' ).    " oo_result is
    returned for testing purpose "
    MESSAGE 'Allocated an Elastic IP address' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = "|{" lo_exception->av_err_code }" - { lo_exception-
    >av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
```

```
ENDTRY.
```

- For API details, see [AllocateAddress in AWS SDK for SAP ABAP API reference](#).

## Associate an Elastic IP address with an instance

The following code example shows how to associate an Elastic IP address with an Amazon EC2 instance.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_ec2->associateaddress(                                     " oo_result is  
    returned for testing purpose "  
        iv_allocationid = iv_allocation_id  
        iv_instanceid = iv_instance_id  
    ).  
    MESSAGE 'Associated Elastic IP address with an EC2 instance' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        DATA(lv_error) = |" { lo_exception->av_err_code }" - { lo_exception-  
        >av_err_msg }|.  
        MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [AssociateAddress in AWS SDK for SAP ABAP API reference](#).

## Create a security group

The following code example shows how to create an Amazon EC2 security group.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_ec2->createsecuritygroup(                                     " oo_result is  
    returned for testing purpose "  
        iv_description = 'Security group example'  
        iv_groupname = iv_security_group_name  
        iv_vpcid = iv_vpc_id  
    ).  
    MESSAGE 'Security group created' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
```

```
DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.
  MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateSecurityGroup](#) in *AWS SDK for SAP ABAP API reference*.

## Create a security key pair

The following code example shows how to create a security key pair for Amazon EC2.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  oo_result = lo_ec2->createkeypair( iv_keyname = iv_key_name ).
    " oo_result is returned for testing purpose "
  MESSAGE 'EC2 key pair created' TYPE 'I'.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.
      MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateKeyPair](#) in *AWS SDK for SAP ABAP API reference*.

## Create and run an instance

The following code example shows how to create and run an Amazon EC2 instance.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
" Create tags for resource created during instance launch "
DATA lt_tagspecifications TYPE /aws1/
cl_ec2tagspecification=>tt_tagspecificationlist.
  DATA ls_tagspecifications LIKE LINE OF lt_tagspecifications.
  ls_tagspecifications = NEW /aws1/cl_ec2tagspecification(
    iv_resourcetype = 'instance'
    it_tags = VALUE /aws1/cl_ec2tag=>tt_taglist(
      ( NEW /aws1/cl_ec2tag( iv_key = 'Name' iv_value = iv_tag_value ) )
```

```
)  
).  
APPEND ls_tagspecifications TO lt_tagspecifications.  
  
TRY.  
  " Create/Launch EC2 instance "  
  oo_result = lo_ec2->runinstances(                                     " oo_result is  
returned for testing purpose "  
    iv_imageid = iv_ami_id  
    iv_instancetype = 't2.micro'  
    iv_maxcount = 1  
    iv_mincount = 1  
    it_tagspecifications = lt_tagspecifications  
    iv_subnetid = iv_subnet_id  
  ).  
  MESSAGE 'EC2 instance created' TYPE 'I'.  
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
  ENDTRY.
```

- For API details, see [RunInstances](#) in *AWS SDK for SAP ABAP API reference*.

## Delete a security group

The following code example shows how to delete an Amazon EC2 security group.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
  lo_ec2->deletesecuritygroup( iv_groupid = iv_security_group_id ).  
  MESSAGE 'Security group deleted' TYPE 'I'.  
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
  ENDTRY.
```

- For API details, see [DeleteSecurityGroup](#) in *AWS SDK for SAP ABAP API reference*.

## Delete a security key pair

The following code example shows how to delete an Amazon EC2 security key pair.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_ec2->deletekeypair( iv_keyname = iv_key_name ).  
    MESSAGE 'EC2 key pair deleted' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.  
        MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [DeleteKeyPair](#) in *AWS SDK for SAP ABAP API reference*.

## Describe Availability Zones

The following code example shows how to describe Amazon EC2 Availability Zones.

### SDK for SAP ABAP

**Note**

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_ec2->describeavailabilityzones( ) .  
    "oo_result is returned for testing purpose "  
    DATA(lt_zones) = oo_result->get_availabilityzones( ).  
    MESSAGE 'Retrieved information about availability zone(s)' TYPE 'I'.  
  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.  
        MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [DescribeAvailabilityZones](#) in *AWS SDK for SAP ABAP API reference*.

## Describe Regions

The following code example shows how to describe Amazon EC2 Regions.

### SDK for SAP ABAP

**Note**

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_ec2->describeregions( ) .  
    " oo_result is  
    returned for testing purpose "  
    DATA(lt_regions) = oo_result->get_regions( ).  
    MESSAGE 'Retrieved information about region(s)' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- For API details, see [DescribeRegions](#) in *AWS SDK for SAP ABAP API reference*.

## Describe instances

The following code example shows how to describe Amazon EC2 instances.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_ec2->describeinstances( ) .  
    " oo_result  
    is returned for testing purpose "  
  
    " Retrieving details of EC2 instance(s) "  
    DATA: lv_instance_id      TYPE /aws1/ec2string,  
          lv_status           TYPE /aws1/ec2instancename,  
          lv_instance_type    TYPE /aws1/ec2instancetype,  
          lv_image_id         TYPE /aws1/ec2string.  
    LOOP AT oo_result->get_reservations( ) INTO DATA(lo_reservation).  
        LOOP AT lo_reservation->get_instances( ) INTO DATA(lo_instance).  
            lv_instance_id = lo_instance->get_instanceid( ).  
            lv_status = lo_instance->get_state( )->get_name( ).  
            lv_instance_type = lo_instance->get_instancetype( ).  
            lv_image_id = lo_instance->get_imageid( ).  
        ENDLOOP.  
    ENDLOOP.  
    MESSAGE 'Retrieved information about EC2 instances' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- For API details, see [DescribeInstances](#) in *AWS SDK for SAP ABAP API reference*.

## Enable monitoring

The following code example shows how to enable monitoring for a running Amazon EC2 instance.

## SDK for SAP ABAP

### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lt_instance_ids TYPE /aws1/cl_ec2instidstringlist_w=>tt_instanceidstringlist.  
APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO  
lt_instance_ids.  
  
"Perform Dry Run"  
TRY.  
    " DryRun is set to true to check if we have the required permissions to  
monitor the instance without actually making the request "  
    lo_ec2->monitorinstances(  
        it_instanceids = lt_instance_ids  
        iv_dryrun = abap_true  
    ).  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        " If the error code returned is 'DryRunOperation', it means we have the  
required permissions to monitor this instance "  
        IF lo_exception->av_err_code = 'DryRunOperation'.  
            MESSAGE 'Dry run to enable detailed monitoring completed' TYPE 'I'.  
            " DryRun is set to false to enable detailed monitoring "  
            lo_ec2->monitorinstances(  
                it_instanceids = lt_instance_ids  
                iv_dryrun = abap_false  
            ).  
            MESSAGE 'Detailed monitoring enabled' TYPE 'I'.  
            " If the error code returned is 'UnauthorizedOperation', it means we do not  
have the required permissions to monitor this instance "  
            ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.  
                MESSAGE 'Dry run to enable detailed monitoring failed: User does not have the  
permissions to monitor the instance' TYPE 'E'.  
            ELSE.  
                DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
                >av_err_msg }|.  
                MESSAGE lv_error TYPE 'E'.  
            ENDIF.  
        ENDTRY.
```

- For API details, see [MonitorInstances](#) in *AWS SDK for SAP ABAP API reference*.

## Get data about a security group

The following code example shows how to get data about an Amazon EC2 security group.

## SDK for SAP ABAP

### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    DATA lt_group_ids TYPE /aws1/cl_ec2groupidstrlist_w=>tt_groupidstringlist.  
    APPEND NEW /aws1/cl_ec2groupidstrlist_w( iv_value = iv_group_id ) TO  
    lt_group_ids.  
    oo_result = lo_ec2->describesecuritygroups( it_groupids = lt_group_ids ).  
    " oo_result is returned for testing purpose "  
    DATA(lt_security_groups) = oo_result->get_securitygroups( ).  
    MESSAGE 'Retrieved information about security group(s)' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.   
    MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [DescribeSecurityGroups](#) in *AWS SDK for SAP ABAP API reference*.

## Get details about Elastic IP addresses

The following code example shows how to get details about Elastic IP addresses.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_ec2->describeaddresses( ) .  
    " oo_result  
is returned for testing purpose "  
    DATA(lt_addresses) = oo_result->get_addresses( ).  
    MESSAGE 'Retrieved information about Elastic IP addresses' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.   
    MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [DescribeAddresses](#) in *AWS SDK for SAP ABAP API reference*.

## List security key pairs

The following code example shows how to list Amazon EC2 security key pairs.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
    oo_result = lo_ec2->describekeypairs( ) .                               " oo_result is
returned for testing purpose "
    DATA(lt_key_pairs) = oo_result->get_keypairs( ). 
    MESSAGE 'Retrieved information about key pair(s)' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- For API details, see [DescribeKeyPairs](#) in *AWS SDK for SAP ABAP API reference*.

## Reboot an instance

The following code example shows how to reboot an Amazon EC2 instance.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

DATA lt_instance_ids TYPE /aws1/cl_ec2instidstringlist_w=>tt_instanceidstringlist.
APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
lt_instance_ids.

"Perform Dry Run"
TRY.
    " DryRun is set to true to check if we have the required permissions to
reboot the instance without actually making the request "
    lo_ec2->rebootinstances(
        it_instanceids = lt_instance_ids
        iv_dryrun = abap_true
    ).
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    " If the error code returned is 'DryRunOperation', it means we have the
required permissions to reboot this instance "
    IF lo_exception->av_err_code = 'DryRunOperation'.
        MESSAGE 'Dry run to reboot instance completed' TYPE 'I'.
        " DryRun is set to false to make a reboot request "
        lo_ec2->rebootinstances(
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false
        ).
        MESSAGE 'Instance rebooted' TYPE 'I'.
        " If the error code returned is 'UnauthorizedOperation', it means we do not
have the required permissions to reboot this instance "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
            MESSAGE 'Dry run to reboot instance failed: User does not have the permission
to reboot the instance' TYPE 'E'.
        ELSE.
            DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
            MESSAGE lv_error TYPE 'E'.
        ENDIF.

```

```
ENDTRY.
```

- For API details, see [RebootInstances in AWS SDK for SAP ABAP API reference](#).

## Release an Elastic IP address

The following code example shows how to release an Elastic IP address.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_ec2->releaseaddress( iv_allocationid = iv_allocation_id ).  
    MESSAGE 'Elastic IP address released' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
        >av_err_msg }|.  
        MESSAGE lv_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see [ReleaseAddress in AWS SDK for SAP ABAP API reference](#).

## Start an instance

The following code example shows how to start an Amazon EC2 instance.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lt_instance_ids TYPE /aws1/cl_ec2instidstringlist_w=>tt_instanceidstringlist.  
APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO  
lt_instance_ids.  
  
"Perform Dry Run"  
TRY.  
    " DryRun is set to true to check if we have the required permissions to  
    start the instance without actually making the request "  
    lo_ec2->startinstances(  
        it_instanceids = lt_instance_ids  
        iv_dryrun = abap_true  
    ).
```

```

CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
  " If the error code returned is `DryRunOperation`, it means we have the
  required permissions to start this instance "
  IF lo_exception->av_err_code = 'DryRunOperation'.
    MESSAGE 'Dry run to start instance completed' TYPE 'I'.
    " DryRun is set to false to start instance "
    oo_result = lo_ec2->startinstances(           " oo_result is returned for
testing purpose "
      it_instanceids = lt_instance_ids
      iv_dryrun = abap_false
    ).
    MESSAGE 'Successfully started the EC2 instance' TYPE 'I'.
    " If the error code returned is `UnauthorizedOperation`, it means we do not
have the required permissions to start this instance "
    ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
      MESSAGE 'Dry run to start instance failed: User does not have the permissions
to start the instance' TYPE 'E'.
    ELSE.
      DATA(lv_error) = |"{" lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
      MESSAGE lv_error TYPE 'E'.
    ENDIF.
  ENDTRY.

```

- For API details, see [StartInstances](#) in *AWS SDK for SAP ABAP API reference*.

## Stop an instance

The following code example shows how to stop an Amazon EC2 instance.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

DATA lt_instance_ids TYPE /aws1/cl_ec2instidstringlist_w=>tt_instanceidstringlist.
APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
lt_instance_ids.

"Perform Dry Run"
TRY.
  " DryRun is set to true to check if we have the required permissions to
stop the instance without actually making the request "
  lo_ec2->stopinstances(
    it_instanceids = lt_instance_ids
    iv_dryrun = abap_true
  ).
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    " If the error code returned is `DryRunOperation`, it means we have the
    required permissions to stop this instance "
    IF lo_exception->av_err_code = 'DryRunOperation'.
      MESSAGE 'Dry run to stop instance completed' TYPE 'I'.
      " DryRun is set to false to stop instance "
      oo_result = lo_ec2->stopinstances(           " oo_result is returned for
testing purpose "

```

```
    it_instanceids = lt_instance_ids
    iv_dryrun = abap_false
).
MESSAGE 'Successfully stopped the EC2 instance' TYPE 'I'.
" If the error code returned is 'UnauthorizedOperation', it means we do not
have the required permissions to stop this instance "
ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
MESSAGE 'Dry run to stop instance failed: User does not have the permissions
to stop the instance' TYPE 'E'.
ELSE.
DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
MESSAGE lv_error TYPE 'E'.
ENDIF.
ENDTRY.
```

- For API details, see [StopInstances](#) in *AWS SDK for SAP ABAP API reference*.

## Kinesis examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon Kinesis.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4499\)](#)
- [Scenarios \(p. 4503\)](#)

## Actions

### Create a stream

The following code example shows how to create a Kinesis stream.

#### SDK for SAP ABAP

##### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
lo_kns->createstream(
    iv_streamname = iv_stream_name
    iv_shardcount = iv_shard_count
).
MESSAGE 'Stream created' TYPE 'I'.
CATCH /aws1/cx_knsinvalidargumentex.
MESSAGE 'The specified argument was not valid.' TYPE 'E'.
```

```
CATCH /aws1/cx_knslimitexceededex .
MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
CATCH /aws1/cx_knsresourceinuseex .
MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateStream](#) in *AWS SDK for SAP ABAP API reference*.

## Delete a stream

The following code example shows how to delete a Kinesis stream.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  lo_kns->deletestream(
    iv_streamname = iv_stream_name
  ).
  MESSAGE 'Stream deleted' TYPE 'I'.
CATCH /aws1/cx_knslimitexceededex .
  MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
CATCH /aws1/cx_knsresourceinuseex .
  MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
ENDTRY.
```

- For API details, see [DeleteStream](#) in *AWS SDK for SAP ABAP API reference*.

## Describe a stream

The following code example shows how to describe a Kinesis stream.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  oo_result = lo_kns->describestream(
    iv_streamname = iv_stream_name
```

```
).
DATA(lt_stream_description) = oo_result->get_streamdescription( ).
MESSAGE 'Streams retrieved' TYPE 'I'.
CATCH /aws1/cx_knslimitexceededdex .
MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
CATCH /aws1/cx_knsresourcenotfounddex .
MESSAGE 'Resource being access is not found.' TYPE 'E'.
ENDTRY.
```

- For API details, see [DescribeStream in AWS SDK for SAP ABAP API reference](#).

## Get data in batches from a stream

The following code example shows how to get data in batches from a Kinesis stream.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_kns->getrecords(                                     " oo_result is returned for testing
purpose "
        iv_sharditerator = iv_shard_iterator
    ).
    DATA(lt_records) = oo_result->get_records( ).
    MESSAGE 'Record retrieved' TYPE 'I'.
    CATCH /aws1/cx_knsexpirediteratorex .
    MESSAGE 'Iterator expired.' TYPE 'E'.
    CATCH /aws1/cx_knsinvalidargumentex .
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    CATCH /aws1/cx_knskmsaccessdeniedex .
    MESSAGE 'You do not have permission to perform this KMS action.' TYPE 'E'.
    CATCH /aws1/cx_knskmsdisabledex .
    MESSAGE 'KMS key used is disabled' TYPE 'E'.
    CATCH /aws1/cx_knskmsinvalidstateex .
    MESSAGE 'KMS key used is in an invalid state ' TYPE 'E'.
    CATCH /aws1/cx_knskmsnotfoundex .
    MESSAGE 'KMS key used is not found' TYPE 'E'.
    CATCH /aws1/cx_knskmsoptionrequired .
    MESSAGE 'KMS key option is required' TYPE 'E'.
    CATCH /aws1/cx_knskmsthrottlingex .
    MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.' TYPE
'E'.
    CATCH /aws1/cx_knsprovthruputexdex .
    MESSAGE 'The request rate for the stream is too high, or the requested data is
too large for the available throughput.' TYPE 'E'.
    CATCH /aws1/cx_knsresourcenotfounddex .
    MESSAGE 'Resource being access is not found.' TYPE 'E'.
ENDTRY.
```

- For API details, see the following topics in [AWS SDK for SAP ABAP API reference](#).
  - [GetRecords](#)

- [GetShardIterator](#)

## List streams

The following code example shows how to list information about one or more Kinesis streams.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_kns->liststreams(           " oo_result is returned for testing  
purpose "  
        "set Limit to specify that a maximum of streams should be returned"  
        iv_limit = iv_limit  
    ).  
    DATA(lt_streams) = oo_result->get_streamnames( ).  
    MESSAGE 'Streams listed' TYPE 'I'.  
    CATCH /aws1/cx_knslimitexceededex .  
        MESSAGE 'The request processing has failed because of a limit exceed  
exception.' TYPE 'E'.  
    ENDTRY.
```

- For API details, see [ListStreams](#) in *AWS SDK for SAP ABAP API reference*.

## Put data into a stream

The following code example shows how to put data into a Kinesis stream.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_kns->putrecord(           " oo_result is returned for testing  
purpose "  
        iv_streamname = iv_stream_name  
        iv_data      = iv_data  
        iv_partitionkey = iv_partition_key  
    ).  
    MESSAGE 'Record created' TYPE 'I'.  
    CATCH /aws1/cx_knsinvalidargumentex .  
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.  
    CATCH /aws1/cx_knskmsaccessdeniedex .  
        MESSAGE 'You do not have permission to perform this KMS action.' TYPE 'E'.
```

```
CATCH /aws1/cx_knskmsdisabledex .
    MESSAGE 'KMS key used is disabled' TYPE 'E'.
CATCH /aws1/cx_knskmsinvalidstateex .
    MESSAGE 'KMS key used is in an invalid state' TYPE 'E'.
CATCH /aws1/cx_knskmsnotfoundex .
    MESSAGE 'KMS key used is not found' TYPE 'E'.
CATCH /aws1/cx_knskmsoptionrequired .
    MESSAGE 'KMS key option is required' TYPE 'E'.
CATCH /aws1/cx_knskmsthrottlingex .
    MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.' TYPE 'E'.
CATCH /aws1/cx_knsprovthruputexcex .
    MESSAGE 'The request rate for the stream is too high, or the requested data is
too large for the available throughput.' TYPE 'E'.
CATCH /aws1/cx_knsresourcenotfoundex .
    MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
ENDTRY.
```

- For API details, see [PutRecord](#) in *AWS SDK for SAP ABAP API reference*.

## Register a consumer

The following code example shows how to register a consumer to a Kinesis stream.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_kns->registerstreamconsumer(          " oo_result is returned for
testing purpose "
        iv_streamarn = iv_stream_arn
        iv_consumername = iv_consumer_name
    ).
    MESSAGE 'Stream consumer registered' TYPE 'I'.
CATCH /aws1/cx_knsinvalidargumentex .
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcelimitexd.
    MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
CATCH /aws1/cx_sgmresourceinuse.
    MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcenotfound.
    MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
ENDTRY.
```

- For API details, see [RegisterStreamConsumer](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios

### Get started with data streams

The following code example shows how to:

- Create a stream.
- Put record in a stream.
- Create a shard iterator.
- Read the record.
- Delete stream.

## SDK for SAP ABAP

### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_stream_describe_result TYPE REF TO /aws1/cl_knsdescriostreamoutput.
DATA lo_stream_description TYPE REF TO /aws1/cl_knsstreamdescription.
DATA lo_sharditerator TYPE REF TO /aws1/cl_knsgetsharditerator01.
DATA lo_record_result TYPE REF TO /aws1/cl_knsputrecordoutput.

"Create stream"
TRY.
    lo_kns->creatostream(
        iv_streamname = iv_stream_name
        iv_shardcount = iv_shard_count
    ).
    MESSAGE 'Stream created' TYPE 'I'.
    CATCH /aws1/cx_knsinvalidargumentex.
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    CATCH /aws1/cx_knslimitexceededex .
        MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
    CATCH /aws1/cx_knsresourceinuseex .
        MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
ENDTRY.

"Wait for steam to becomes active"
lo_stream_describe_result = lo_kns->describestream( iv_streamname =
iv_stream_name ).
lo_stream_description = lo_stream_describe_result->get_streamdescription( ).
WHILE lo_stream_description->get_streamstatus( ) <> 'ACTIVE'.
    IF sy-index = 30.
        EXIT.                      "maximum 5 minutes"
    ENDIF.
    WAIT UP TO 10 SECONDS.
    lo_stream_describe_result = lo_kns->describestream( iv_streamname =
iv_stream_name ).
    lo_stream_description = lo_stream_describe_result->get_streamdescription( ).
ENDWHILE.

"Create record"
TRY.
    lo_record_result = lo_kns->putrecord(
        iv_streamname = iv_stream_name
        iv_data       = iv_data
        iv_partitionkey = iv_partition_key
    ).
    MESSAGE 'Record created' TYPE 'I'.
```

```

CATCH /aws1/cx_knsinvalidargumentex .
    MESSAGE 'The specified argument was not valid.' TYPE 'E'.
CATCH /aws1/cx_knskmsaccessdeniedex .
    MESSAGE 'You do not have permission to perform this KMS action.' TYPE 'E'.
CATCH /aws1/cx_knskmsdisabledex .
    MESSAGE 'KMS key used is disabled' TYPE 'E'.
CATCH /aws1/cx_knskmsinvalidstateex .
    MESSAGE 'KMS key used is in an invalid state ' TYPE 'E'.
CATCH /aws1/cx_knskmsnotfoundex .
    MESSAGE 'KMS key used is not found' TYPE 'E'.
CATCH /aws1/cx_knskmsoptinrequired .
    MESSAGE 'KMS key option is required' TYPE 'E'.
CATCH /aws1/cx_knskmsthrottlingex .
    MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.' TYPE
'E'.
    CATCH /aws1/cx_knsprovthruputexcdex .
        MESSAGE 'The request rate for the stream is too high, or the requested data is
too large for the available throughput.' TYPE 'E'.
        CATCH /aws1/cx_knsresourcenotfoundex .
            MESSAGE 'Resource being access is not found.' TYPE 'E'.
    ENDTRY.

"Create a shard iterator in order to read the record"
TRY.
    lo_sharditerator = lo_kns->getsharditerator(
        iv_shardid = lo_record_result->get_shardid( )
        iv_sharditeratortype = iv_sharditeratortype
        iv_streamname = iv_stream_name
    ).
    MESSAGE 'Shard iterator created' TYPE 'I'.
    CATCH /aws1/cx_knsinvalidargumentex .
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    CATCH /aws1/cx_knsprovthruputexcdex .
        MESSAGE 'The request rate for the stream is too high, or the requested data is
too large for the available throughput.' TYPE 'E'.
        CATCH /aws1/cx_sgmresourcenotfound .
            MESSAGE 'Resource being access is not found.' TYPE 'E'.
    ENDTRY.

"Read the record"
TRY.
    oo_result = lo_kns->getrecords(                                     " oo_result is returned for
testing purpose "
        iv_sharditerator      = lo_sharditerator->get_sharditerator( )
    ).
    MESSAGE 'Shard iterator created' TYPE 'I'.
    CATCH /aws1/cx_knsexpirediteratorex .
        MESSAGE 'Iterator expired.' TYPE 'E'.
    CATCH /aws1/cx_knsinvalidargumentex .
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    CATCH /aws1/cx_knskmsaccessdeniedex .
        MESSAGE 'You do not have permission to perform this KMS action.' TYPE 'E'.
    CATCH /aws1/cx_knskmsdisabledex .
        MESSAGE 'KMS key used is disabled' TYPE 'E'.
    CATCH /aws1/cx_knskmsinvalidstateex .
        MESSAGE 'KMS key used is in an invalid state ' TYPE 'E'.
    CATCH /aws1/cx_knskmsnotfoundex .
        MESSAGE 'KMS key used is not found' TYPE 'E'.
    CATCH /aws1/cx_knskmsoptinrequired .
        MESSAGE 'KMS key option is required' TYPE 'E'.
    CATCH /aws1/cx_knskmsthrottlingex .
        MESSAGE 'The rate of requests to AWS KMS is exceeding the request quotas.' TYPE
'E'.
        CATCH /aws1/cx_knsprovthruputexcdex .
            MESSAGE 'The request rate for the stream is too high, or the requested data is
too large for the available throughput.' TYPE 'E'.

```

```
CATCH /aws1/cx_knsresourcenotfoundex .
  MESSAGE 'Resource being access is not found.' TYPE 'E'.
ENDTRY.

"Delete Stream"
TRY.
  lo_kns->deletestream(
    iv_streamname = iv_stream_name
  ).
  MESSAGE 'Stream deleted' TYPE 'I'.
CATCH /aws1/cx_knslimitexceededex .
  MESSAGE 'The request processing has failed because of a limit exceed
exception.' TYPE 'E'.
CATCH /aws1/cx_knsresourceinuseex .
  MESSAGE 'The request processing has failed because the resource is in use.'
TYPE 'E'.
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [CreateStream](#)
  - [DeleteStream](#)
  - [GetRecords](#)
  - [GetShardIterator](#)
  - [PutRecord](#)

## Lambda examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with AWS Lambda.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4506\)](#)
- [Scenarios \(p. 4511\)](#)

## Actions

### Create a function

The following code example shows how to create a Lambda function.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
```

```
lo_lmd->createfunction(
    iv_functionname = iv_function_name
    iv_runtime = `python3.9`
    iv_role = iv_role_arn
    iv_handler = iv_handler
    io_code = io_zip_file
    iv_description = 'AWS Lambda code example'
).
MESSAGE 'Lambda function created' TYPE 'I'.
CATCH /aws1/cx_lmdcodesigningcfgno00.
MESSAGE 'Code signing configuration does not exist' TYPE 'E'.
CATCH /aws1/cx_lmdcodestorageexcdex.
MESSAGE 'Maximum total code size per account exceeded' TYPE 'E'.
CATCH /aws1/cx_lmdcodeverification00.
MESSAGE 'Code signature failed one or more validation checks for signature
mismatch or expiration' TYPE 'E'.
CATCH /aws1/cx_lmdinvalidcodesigex.
MESSAGE 'Code signature failed the integrity check' TYPE 'E'.
CATCH /aws1/cx_lmdinvparamvalueeex.
MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
CATCH /aws1/cx_lmdresourceconflictex.
MESSAGE 'Resource already exists or another operation is in progress' TYPE 'E'.
CATCH /aws1/cx_lmdresourcenotfoundex.
MESSAGE 'The requested resource does not exist' TYPE 'E'.
CATCH /aws1/cx_lmdserviceexception.
MESSAGE 'An internal problem was encountered by the AWS Lambda service.' TYPE
'E'.
CATCH /aws1/cx_lmdtoomanyrequestsex.
MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateFunction](#) in *AWS SDK for SAP ABAP API reference*.

## Delete a function

The following code example shows how to delete a Lambda function.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    lo_lmd->deletefunction( iv_functionname = iv_function_name ).
    MESSAGE 'Lambda function deleted' TYPE 'I'.
CATCH /aws1/cx_lmdinvparamvalueeex.
MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
CATCH /aws1/cx_lmdresourceconflictex.
MESSAGE 'Rresource already exists or another operation is in progress' TYPE
'E'.
CATCH /aws1/cx_lmdresourcenotfoundex.
MESSAGE 'The requested resource does not exist' TYPE 'E'.
CATCH /aws1/cx_lmdserviceexception.
MESSAGE 'An internal problem was encountered by the AWS Lambda service.' TYPE
'E'.
CATCH /aws1/cx_lmdtoomanyrequestsex.
```

```
MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
ENDTRY.
```

- For API details, see [DeleteFunction](#) in *AWS SDK for SAP ABAP API reference*.

## Get a function

The following code example shows how to get a Lambda function.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_lmd->getfunction( iv_functionname = iv_function_name ).      "
oo_result is returned for testing purpose "
    MESSAGE 'Lambda function information retrieved' TYPE 'I'.
    CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
    CATCH /aws1/cx_lmdserviceexception.
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.' TYPE
'E'.
    CATCH /aws1/cx_lmdtoomanyrequestsex.
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
ENDTRY.
```

- For API details, see [GetFunction](#) in *AWS SDK for SAP ABAP API reference*.

## Invoke a function

The following code example shows how to invoke a Lambda function.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    DATA(lv_json) = /aws1/cl_rt_util=>string_to_xstring(
        `` &&
        ` "action": "increment", ` &&
        ` "number": 10` &&
        `}`
    ).

```

```

    oo_result = lo_lmd->invoke(                               " oo_result is returned for
testing purpose "
        iv_functionname = iv_function_name
        iv_payload = lv_json
    ).
    MESSAGE 'Lambda function invoked' TYPE 'I'.
CATCH /aws1/cx_lmdinvparamvalueex.
    MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
CATCH /aws1/cx_lmdinvrequestcontext.
    MESSAGE 'Unable to parse request body as JSON' TYPE 'E'.
CATCH /aws1/cx_lmdinvalidzipfileex.
    MESSAGE 'The deployment package could not be unzipped' TYPE 'E'.
CATCH /aws1/cx_lmdrequesttoolargeex.
    MESSAGE 'Invoke request body JSON input limit was exceeded by the request
payload.' TYPE 'E'.
CATCH /aws1/cx_lmdresourceconflictex.
    MESSAGE 'Rresource already exists or another operation is in progress' TYPE
'E'.
CATCH /aws1/cx_lmdresourcenotfoundex.
    MESSAGE 'The requested resource does not exist' TYPE 'E'.
CATCH /aws1/cx_lmdserviceexception.
    MESSAGE 'An internal problem was encountered by the AWS Lambda service' TYPE
'E'.
CATCH /aws1/cx_lmdtoomanyrequestsex.
    MESSAGE 'The maximum request throughput was reached' TYPE 'E'.
CATCH /aws1/cx_lmdunsuppedmediatyp00.
    MESSAGE 'Invoke request body does not have JSON as its content type' TYPE 'E'.
ENDTRY.
```

- For API details, see [Invoke](#) in *AWS SDK for SAP ABAP API reference*.

## List functions

The following code example shows how to list Lambda functions.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

TRY.
    oo_result = lo_lmd->listfunctions( ).           " oo_result is returned for
testing purpose "
        DATA(lt_functions) = oo_result->get_functions( ).
        MESSAGE 'Retrieved list of lambda function(s)' TYPE 'I'.
CATCH /aws1/cx_lmdinvparamvalueex.
    MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
CATCH /aws1/cx_lmdserviceexception.
    MESSAGE 'An internal problem was encountered by the AWS Lambda service.' TYPE
'E'.
CATCH /aws1/cx_lmdtoomanyrequestsex.
    MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.
ENDTRY.
```

- For API details, see [ListFunctions](#) in *AWS SDK for SAP ABAP API reference*.

## Update function code

The following code example shows how to update Lambda function code.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_lmd->updatefunctioncode(      " oo_result is returned for testing  
purpose "  
        iv_functionname = iv_function_name  
        iv_zipfile = io_zip_file  
    ).  
  
    MESSAGE 'Lambda function code updated' TYPE 'I'.  
    CATCH /aws1/cx_lmdcodesigningcfgno00.  
        MESSAGE 'Code signing configuration does not exist' TYPE 'E'.  
    CATCH /aws1/cx_lmdcodestorageexcex.  
        MESSAGE 'Maximum total code size per account exceeded' TYPE 'E'.  
    CATCH /aws1/cx_lmdcodeverification00.  
        MESSAGE 'Code signature failed one or more validation checks for signature  
mismatch or expiration' TYPE 'E'.  
    CATCH /aws1/cx_lmdinvalidcodesigex.  
        MESSAGE 'Code signature failed the integrity check' TYPE 'E'.  
    CATCH /aws1/cx_lmdinvparamvalueex.  
        MESSAGE 'The request contains an invalid parameter' TYPE 'E'.  
    CATCH /aws1/cx_lmdresourceconflictex.  
        MESSAGE 'Rresource already exists or another operation is in progress' TYPE  
'E'.  
    CATCH /aws1/cx_lmdresourcenotfoundex.  
        MESSAGE 'The requested resource does not exist' TYPE 'E'.  
    CATCH /aws1/cx_lmdserviceexception.  
        MESSAGE 'An internal problem was encountered by the AWS Lambda service.' TYPE  
'E'.  
    CATCH /aws1/cx_lmdtoomanyrequestsex.  
        MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.  
ENDTRY.
```

- For API details, see [UpdateFunctionCode](#) in *AWS SDK for SAP ABAP API reference*.

## Update function configuration

The following code example shows how to update Lambda function configuration.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_lmd->updatefunctionconfiguration(      " oo_result is returned  
for testing purpose "  
        iv_functionname = iv_function_name  
        iv_runtime = iv_runtime  
        iv_description = 'Updated Lambda Function'  
        iv_memorysize = iv_memory_size  
    ).  
  
    MESSAGE 'Lambda function configuration/settings updated' TYPE 'I'.  
    CATCH /aws1/cx_lmdcodesigningcfgno00.  
        MESSAGE 'Code signing configuration does not exist' TYPE 'E'.  
        CATCH /aws1/cx_lmdcodeverification00.  
            MESSAGE 'Code signature failed one or more validation checks for signature  
mismatch or expiration' TYPE 'E'.  
            CATCH /aws1/cx_lmdinvalidcodesigex.  
                MESSAGE 'Code signature failed the integrity check' TYPE 'E'.  
                CATCH /aws1/cx_lmdinvparamvalueex.  
                    MESSAGE 'The request contains an invalid parameter' TYPE 'E'.  
                    CATCH /aws1/cx_lmdresourceconflictex.  
                        MESSAGE 'Rresource already exists or another operation is in progress' TYPE  
'E'.  
                        CATCH /aws1/cx_lmdresourcenotfoundex.  
                            MESSAGE 'The requested resource does not exist' TYPE 'E'.  
                            CATCH /aws1/cx_lmdserviceexception.  
                                MESSAGE 'An internal problem was encountered by the AWS Lambda service.' TYPE  
'E'.  
                                CATCH /aws1/cx_lmdtoomanyrequestsex.  
                                    MESSAGE 'The maximum request throughput was reached.' TYPE 'E'.  
    ENDTRY.
```

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios

### Get started with functions

The following code example shows how to:

- Create an AWS Identity and Access Management (IAM) role that grants Lambda permission to write to logs.
- Create a Lambda function and upload handler code.
- Invoke the function with a single parameter and get results.
- Update the function code and configure its Lambda environment with an environment variable.
- Invoke the function with new parameters and get results. Display the execution log that's returned from the invocation.
- List the functions for your account.
- Delete the IAM role and the Lambda function.

For more information, see [Create a Lambda function with the console](#).

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    "Create an AWS Identity and Access Management (IAM) role that grants Lambda  
permission to write to logs"  
    DATA(lv_policy_document) = `{' &&  
        `"Version":"2012-10-17",` &&  
        `"Statement": [` &&  
            `{` &&  
                `'"Effect": "Allow",` &&  
                `'"Action": [` &&  
                    `'"sts:AssumeRole"' &&  
                `],` &&  
                `'"Principal": {` &&  
                    `'"Service": [` &&  
                        `'"lambda.amazonaws.com"' &&  
                    `],` &&  
                `}` &&  
            `]` &&  
        `]` &&  
    `}`.  
TRY.  
    DATA(lo_create_role_output) = lo_iam->createrole(  
        iv_rolename = iv_role_name  
        iv_assumerolepolicydocument = lv_policy_document  
        iv_description = 'Grant lambda permission to write to logs'  
    ).  
MESSAGE 'IAM Role created' TYPE 'I'.  
WAIT UP TO 10 SECONDS.           " Just to make sure IAM role is ready for  
use "  
CATCH /aws1/cx_iamentityalrdyexec.  
    MESSAGE 'IAM role already exist' TYPE 'E'.  
CATCH /aws1/cx_iaminvalidinputex.  
    MESSAGE 'The request contains an invalid parameter' TYPE 'E'.  
CATCH /aws1/cx_iammalformedplydocex.  
    MESSAGE 'Policy document in the request is malformed' TYPE 'E'.  
ENDTRY.  
  
TRY.  
    lo_iam->attachrolepolicy(  
        iv_rolename = iv_role_name  
        iv_policyarn = 'arn:aws:iam::aws:policy/service-role/  
AWSLambdaBasicExecutionRole'  
    ).  
    MESSAGE 'Attached policy to the IAM role' TYPE 'I'.  
CATCH /aws1/cx_iaminvalidinputex.  
    MESSAGE 'The request contains an invalid parameter' TYPE 'E'.  
CATCH /aws1/cx_iamnosuchentityex.  
    MESSAGE 'The requested resource entity does not exist' TYPE 'E'.  
CATCH /aws1/cx_iamplynottachableex.  
    MESSAGE 'Service role policies can only be attached to the service-linked  
role for that service' TYPE 'E'.  
CATCH /aws1/cx_iamunmodableentityex.  
    MESSAGE 'Service that depends on the service-linked role is not modifiable'  
TYPE 'E'.  
ENDTRY.  
  
" Create a Lambda function and upload handler code."  
" Lambda function performs 'increment' action on a number "  
TRY.  
    lo_lmd->createfunction(
```

```

        iv_functionname = iv_function_name
        iv_runtime = `python3.9'
        iv_role = lo_create_role_output->get_role( )->get_arn( )
        iv_handler = iv_handler
        io_code = io_initial_zip_file
        iv_description = 'AWS Lambda code example'
    ).
    MESSAGE 'Lambda function created' TYPE 'I'.
    CATCH /aws1/cx_lmdcodestorageexcdex.
        MESSAGE 'Maximum total code size per account exceeded' TYPE 'E'.
    CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
    CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist' TYPE 'E'.
    ENDTRY.

    " Verify the function is in Active state "
    WHILE lo_lmd->getfunction( iv_functionname = iv_function_name )-
>get_configuration( )->ask_state( ) <> 'Active'.
        IF sy-index = 10.
            EXIT.                      " max 10 seconds "
        ENDIF.
        WAIT UP TO 1 SECONDS.
    ENDWHILE.

    "Invoke the function with a single parameter and get results."
    TRY.
        DATA(lv_json) = /aws1/cl_rt_util=>string_to_xstring(
            `{
                &&
                `action": "increment", ` &&
                `number": 10` &&
            }`
        ).
        DATA(lo_initial_invoke_output) =  lo_lmd->invoke(
            iv_functionname = iv_function_name
            iv_payload = lv_json
        ).
        ov_initial_invoke_payload = lo_initial_invoke_output->get_payload( ).
    " ov_initial_invoke_payload is returned for testing purpose "
        DATA(lo_writer_json) = cl_sxml_string_writer=>create( type =
if_sxml=>co_xt_json ).
        CALL TRANSFORMATION id SOURCE XML ov_initial_invoke_payload RESULT XML
lo_writer_json.
        DATA(lv_result) = cl_abap_codepage=>convert_from( lo_writer_json-
>get_output( ) ).
        MESSAGE 'Lambda function invoked' TYPE 'I'.
        CATCH /aws1/cx_lmdinvparamvalueex.
            MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
        CATCH /aws1/cx_lmdinvrequestcontex.
            MESSAGE 'Unable to parse request body as JSON' TYPE 'E'.
        CATCH /aws1/cx_lmdresourcenotfoundex.
            MESSAGE 'The requested resource does not exist' TYPE 'E'.
        CATCH /aws1/cx_lmdunsuppemediatyp00.
            MESSAGE 'Invoke request body does not have JSON as its content type' TYPE
'E'.
        ENDTRY.

        " Update the function code and configure its Lambda environment with an
environment variable. "
        " Lambda function is updated to perform 'decrement' action as well "
        TRY.
            lo_lmd->updatefunctioncode(
                iv_functionname = iv_function_name
                iv_zipfile = io_updated_zip_file
            ).
            WAIT UP TO 10 SECONDS.           " Just to make sure update is completed "

```

```

MESSAGE 'Lambda function code updated' TYPE 'I'.
CATCH /aws1/cx_lmdcodestorageexcex.
MESSAGE 'Maximum total code size per account exceeded' TYPE 'E'.
CATCH /aws1/cx_lmdinvparamvalueex.
MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
CATCH /aws1/cx_lmdresourcenotfoundex.
MESSAGE 'The requested resource does not exist' TYPE 'E'.
ENDTRY.

TRY.
  DATA lt_variables TYPE /aws1/
cl_lmdenvironmentvaria00=>tt_environmentvariables.
  DATA ls_variable LIKE LINE OF lt_variables.
  ls_variable-key = 'LOG_LEVEL'.
  ls_variable-value = NEW /aws1/cl_lmdenvironmentvaria00( iv_value =
'info' ).
  INSERT ls_variable INTO TABLE lt_variables.

  lo_lmd->updatefunctionconfiguration(
    iv_functionname = iv_function_name
    io_environment = NEW /aws1/cl_lmdenvironment( it_variables =
lt_variables )
  ).
  WAIT UP TO 10 SECONDS.           " Just to make sure update is completed "
  MESSAGE 'Lambda function configuration/settings updated' TYPE 'I'.
CATCH /aws1/cx_lmdinvparamvalueex.
MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
CATCH /aws1/cx_lmdresourceconflictex.
MESSAGE 'Rresource already exists or another operation is in progress' TYPE
'E'.
CATCH /aws1/cx_lmdresourcenotfoundex.
MESSAGE 'The requested resource does not exist' TYPE 'E'.
ENDTRY.

"Invoke the function with new parameters and get results. Display the execution
log that's returned from the invocation."
TRY.
  lv_json = /aws1/cl_rt_util=>string_to_xstring(
`{` &&
`  "action": "decrement", ` &&
`  "number": 10` &&
`}`
).
  DATA(lo_updated_invoke_output) = lo_lmd->invoke(
    iv_functionname = iv_function_name
    iv_payload = lv_json
  ).
  ov_updated_invoke_payload = lo_updated_invoke_output->get_payload( ).
" ov_updated_invoke_payload is returned for testing purpose "
  lo_writer_json = cl_sxml_string_writer=>create( type =
if_sxml=>co_xt_json ).
  CALL TRANSFORMATION id SOURCE XML ov_updated_invoke_payload RESULT XML
lo_writer_json.
  lv_result = cl_abap_codepage=>convert_from( lo_writer_json-
>get_output( )).
  MESSAGE 'Lambda function invoked' TYPE 'I'.
CATCH /aws1/cx_lmdinvparamvalueex.
MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
CATCH /aws1/cx_lmdinvrequestcontex.
MESSAGE 'Unable to parse request body as JSON' TYPE 'E'.
CATCH /aws1/cx_lmdresourcenotfoundex.
MESSAGE 'The requested resource does not exist' TYPE 'E'.
CATCH /aws1/cx_lmdunsuppedmediatyp00.
MESSAGE 'Invoke request body does not have JSON as its content type' TYPE
'E'.
ENDTRY.

```

```

" List the functions for your account. "
TRY.
    DATA(lo_list_output) = lo_lmd->listfunctions( ).
    DATA(lt_functions) = lo_list_output->get_functions( ).
    MESSAGE 'Retrieved list of lambda function(s)' TYPE 'I'.
    CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
    ENDTRY.

" Delete the Lambda function. "
TRY.
    lo_lmd->deletefunction( iv_functionname = iv_function_name ).
    MESSAGE 'Lambda function deleted' TYPE 'I'.
    CATCH /aws1/cx_lmdinvparamvalueex.
        MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
    CATCH /aws1/cx_lmdresourcenotfoundex.
        MESSAGE 'The requested resource does not exist' TYPE 'E'.
    ENDTRY.

" Detach role policy "
TRY.
    lo_iam->detachrolepolicy(
        iv_rolename  = iv_role_name
        iv_policyarn = 'arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole'
    ).
    MESSAGE 'Detached policy to the IAM role' TYPE 'I'.
    CATCH /aws1/cx_iaminvalidinputex.
        MESSAGE 'The request contains an invalid parameter' TYPE 'E'.
    CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'The requested resource entity does not exist' TYPE 'E'.
    CATCH /aws1/cx_iamplynottachableex.
        MESSAGE 'Service role policies can only be attached to the service-linked
role for that service' TYPE 'E'.
    CATCH /aws1/cx_iamunmodableentityex.
        MESSAGE 'Service that depends on the service-linked role is not modifiable'
TYPE 'E'.
    ENDTRY.

" Delete the IAM role. "
TRY.
    lo_iam->deleterole( iv_rolename = iv_role_name ).
    MESSAGE 'IAM role deleted' TYPE 'I'.
    CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'The requested resource entity does not exist' TYPE 'E'.
    CATCH /aws1/cx_iamunmodableentityex.
        MESSAGE 'Service that depends on the service-linked role is not modifiable'
TYPE 'E'.
    ENDTRY.

CATCH /aws1/cx_rt_service_generic INTO lo_exception.
    DATA(lv_error) = lo_exception->get_longtext( ).
    MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.

- [CreateFunction](#)
- [DeleteFunction](#)
- [GetFunction](#)
- [Invoke](#)
- [ListFunctions](#)

- [UpdateFunctionCode](#)
- [UpdateFunctionConfiguration](#)

## Amazon S3 examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon Simple Storage Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4516\)](#)
- [Scenarios \(p. 4519\)](#)

## Actions

### [Copy an object from one bucket to another](#)

The following code example shows how to copy an S3 object from one bucket to another.

#### SDK for SAP ABAP

##### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_s3->copyobject(  
        iv_bucket = iv_dest_bucket  
        iv_key = iv_dest_object  
        iv_copysource = |{ iv_src_bucket }/{ iv_src_object }|  
    ).  
    MESSAGE 'Object copied to another bucket' TYPE 'I'.  
    CATCH /aws1/cx_s3_nosuchbucket.  
        MESSAGE 'Bucket does not exist' TYPE 'E'.  
    CATCH /aws1/cx_s3_nosuchkey.  
        MESSAGE 'Object key does not exist' TYPE 'E'.  
    ENDTRY.
```

- For API details, see [CopyObject](#) in *AWS SDK for SAP ABAP API reference*.

## Create a bucket

The following code example shows how to create an S3 bucket.

#### SDK for SAP ABAP

##### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_s3->createbucket(  
        iv_bucket = iv_bucket_name  
    ).  
    MESSAGE 'S3 bucket created' TYPE 'I'.  
CATCH /aws1/cx_s3_bucketalrdyexists.  
    MESSAGE 'Bucket name already exists' TYPE 'E'.  
CATCH /aws1/cx_s3_bktalrdyownedbyyou.  
    MESSAGE 'Bucket already exist and is owned by you' TYPE 'E'.  
ENDTRY.
```

- For API details, see [CreateBucket](#) in *AWS SDK for SAP ABAP API reference*.

## Delete an empty bucket

The following code example shows how to delete an empty S3 bucket.

### SDK for SAP ABAP

**Note**

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_s3->deletebucket(  
        iv_bucket = iv_bucket_name  
    ).  
    MESSAGE 'Deleted S3 bucket' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteBucket](#) in *AWS SDK for SAP ABAP API reference*.

## Delete an object

The following code example shows how to delete an S3 object.

### SDK for SAP ABAP

**Note**

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_s3->deleteobject(  
        iv_bucket = iv_bucket_name  
        iv_key = iv_object_key  
    ).  
    MESSAGE 'Object deleted from S3 bucket' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist' TYPE 'E'.  
ENDTRY.
```

- For API details, see [DeleteObject](#) in *AWS SDK for SAP ABAP API reference*.

## Get an object from a bucket

The following code example shows how to read data from an object in an S3 bucket.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_s3->getobject(          " oo_result is returned for testing  
purpose "  
        iv_bucket = iv_bucket_name  
        iv_key = iv_object_key  
    ).  
    DATA(lv_object_data) = oo_result->get_body( ).  
    MESSAGE 'Object retrieved from S3 bucket' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist' TYPE 'E'.  
CATCH /aws1/cx_s3_nosuchkey.  
    MESSAGE 'Object key does not exist' TYPE 'E'.  
ENDTRY.
```

- For API details, see [GetObject](#) in *AWS SDK for SAP ABAP API reference*.

## List objects in a bucket

The following code example shows how to list objects in an S3 bucket.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_s3->listobjects(           " oo_result is returned for testing  
purpose "  
        iv_bucket = iv_bucket_name  
    ).  
    MESSAGE 'Retrieved list of object(s) in S3 bucket' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist' TYPE 'E'.  
ENDTRY.
```

- For API details, see [ListObjects](#) in *AWS SDK for SAP ABAP API reference*.

## Upload an object to a bucket

The following code example shows how to upload an object to an S3 bucket.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Get contents of file from application server"  
DATA lv_body TYPE xstring.  
OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.  
READ DATASET iv_file_name INTO lv_body.  
CLOSE DATASET iv_file_name.  
  
"Upload/Put an object to S3 bucket"  
TRY.  
    lo_s3->putobject(  
        iv_bucket = iv_bucket_name  
        iv_key = iv_file_name  
        iv_body = lv_body  
    ).  
    MESSAGE 'Object uploaded to S3 bucket' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist' TYPE 'E'.  
ENDTRY.
```

- For API details, see [PutObject](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios

### Get started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.

- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.
- Delete a bucket.

## SDK for SAP ABAP

### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).  
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).  
  
" Create S3 Bucket "  
TRY.  
    lo_s3->createbucket(  
        iv_bucket = iv_bucket_name  
    ).  
    MESSAGE 'S3 bucket created' TYPE 'I'.  
CATCH /aws1/cx_s3_bucketalrdyexists.  
    MESSAGE 'Bucket name already exists' TYPE 'E'.  
CATCH /aws1/cx_s3_bktalrdyownedbyyou.  
    MESSAGE 'Bucket already exist and is owned by you' TYPE 'E'.  
ENDTRY.  
  
"Upload an object to a S3 bucket"  
TRY.  
    "Get contents of file from application server"  
    DATA lv_file_content TYPE xstring.  
    OPEN DATASET iv_key FOR INPUT IN BINARY MODE.  
    READ DATASET iv_key INTO lv_file_content.  
    CLOSE DATASET iv_key.  
  
    lo_s3->putobject(  
        iv_bucket = iv_bucket_name  
        iv_key = iv_key  
        iv_body = lv_file_content  
    ).  
    MESSAGE 'Object uploaded to S3 bucket' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist' TYPE 'E'.  
ENDTRY.  
  
" Get an object from a bucket "  
TRY.  
    DATA(lo_result) = lo_s3->getobject(  
        iv_bucket = iv_bucket_name  
        iv_key = iv_key  
    ).  
    DATA(lv_object_data) = lo_result->get_body( ).  
    MESSAGE 'Object retrieved from S3 bucket' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist' TYPE 'E'.  
CATCH /aws1/cx_s3_nosuchkey.  
    MESSAGE 'Object key does not exist' TYPE 'E'.
```

```

ENDTRY.

" Copy an object to a subfolder in a bucket "
TRY.
    lo_s3->copyobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }|
        iv_copysource = |{ iv_bucket_name }/{ iv_key }|
    ).
    MESSAGE 'Object copied to a subfolder' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist' TYPE 'E'.
    CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist' TYPE 'E'.
ENDTRY.

" List objects in the bucket "
TRY.
    DATA(lo_list) = lo_s3->listobjects(
        iv_bucket = iv_bucket_name
    ).
    MESSAGE 'Retrieved list of object(s) in S3 bucket' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist' TYPE 'E'.
ENDTRY.
DATA text TYPE string VALUE 'Object List - '.
DATA lv_object_key TYPE /aws1/s3_objectkey.
LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
    lv_object_key = lo_object->get_key( ).
    CONCATENATE lv_object_key ', ' INTO text.
ENDLOOP.
MESSAGE text TYPE 'I'.

" Delete the objects in a bucket "
TRY.
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
    ).
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }|
    ).
    MESSAGE 'Object(s) deleted from S3 bucket' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist' TYPE 'E'.
ENDTRY.

" Delete the bucket "
TRY.
    lo_s3->deletebucket(
        iv_bucket = iv_bucket_name
    ).
    MESSAGE 'Deleted S3 bucket' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist' TYPE 'E'.
ENDTRY.

```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)

- [DeleteObjects](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

## Amazon SNS examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon Simple Notification Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4522\)](#)
- [Scenarios \(p. 4526\)](#)

## Actions

### Create a topic

The following code example shows how to create an Amazon SNS topic.

#### SDK for SAP ABAP

##### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result is  
    returned for testing purpose "  
    MESSAGE 'SNS topic created' TYPE 'I'.  
    CATCH /aws1/cx_snstopiclimitecdex.  
        MESSAGE 'Unable to create more topics as you have reached the maximum number of  
        topics allowed.' TYPE 'E'.  
    ENDTRY.
```

- For API details, see [CreateTopic](#) in *AWS SDK for SAP ABAP API reference*.

### Delete a subscription

The following code example shows how to delete an Amazon SNS subscription.

#### SDK for SAP ABAP

##### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).  
    MESSAGE 'Subscription Deleted' TYPE 'I'.  
    CATCH /aws1/cx_snsnotfoundexception.  
        MESSAGE 'Subscription does not exist' TYPE 'E'.  
    CATCH /aws1/cx_snsinvalidparameterex.  
        MESSAGE 'Subscription with "PendingConfirmation" status cannot be deleted/  
unsubscribed, subscription should be confirmed before perform unsubscribe operation'  
        TYPE 'E'.  
    ENDTRY.
```

- For API details, see [Unsubscribe](#) in *AWS SDK for SAP ABAP API reference*.

## Delete a topic

The following code example shows how to delete an Amazon SNS topic and all subscriptions to that topic.

### SDK for SAP ABAP

**Note**

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).  
    MESSAGE 'SNS topic deleted' TYPE 'I'.  
    CATCH /aws1/cx_snsnotfoundexception.  
        MESSAGE 'Topic does not exist' TYPE 'E'.  
    ENDTRY.
```

- For API details, see [DeleteTopic](#) in *AWS SDK for SAP ABAP API reference*.

## Get the properties of a topic

The following code example shows how to get the properties of an Amazon SNS topic.

### SDK for SAP ABAP

**Note**

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sns->gettopicattributes( iv_topicarn = iv_topic_arn ). "  
oo_result is returned for testing purpose "  
    DATA(lt_attributes) = oo_result->get_attributes( ).  
    MESSAGE 'Retrieved attributes/properties of a topic' TYPE 'I'.  
    CATCH /aws1/cx snsnotfoundexception.  
        MESSAGE 'topic does NOT exist' TYPE 'E'.  
    ENDTRY.
```

- For API details, see [GetTopicAttributes](#) in *AWS SDK for SAP ABAP API reference*.

## List the subscribers of a topic

The following code example shows how to retrieve the list of subscribers of an Amazon SNS topic.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sns->listsubscriptions( ).           " oo_result is  
returned for testing purpose "  
    DATA(lt_subscriptions) = oo_result->get_subscriptions( ).  
    MESSAGE 'Retrieved list of subscriber(s)' TYPE 'I'.  
    CATCH /aws1/cx_rt_generic.  
        MESSAGE 'Unable to list subscriber(s)' TYPE 'E'.  
    ENDTRY.
```

- For API details, see [ListSubscriptions](#) in *AWS SDK for SAP ABAP API reference*.

## List topics

The following code example shows how to list Amazon SNS topics.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sns->listtopics( ).           " oo_result is returned for  
testing purpose "  
    DATA(lt_topics) = oo_result->get_topics( ).  
    MESSAGE 'Retrieved list of topic(s)' TYPE 'I'.  
    CATCH /aws1/cx_rt_generic.
```

```
MESSAGE 'Unable to list topic(s)' TYPE 'E'.
ENDTRY.
```

- For API details, see [ListTopics](#) in *AWS SDK for SAP ABAP API reference*.

## Publish to a topic

The following code example shows how to publish messages to an Amazon SNS topic.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_sns->publish(                                " oo_result is returned for testing
purpose "
        iv_topicarn = iv_topic_arn
        iv_message = iv_message
    ).
    MESSAGE 'Message published to SNS topic' TYPE 'I'.
CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist' TYPE 'E'.
ENDTRY.
```

- For API details, see [Publish](#) in *AWS SDK for SAP ABAP API reference*.

## Set topic attributes

The following code example shows how to set Amazon SNS topic attributes.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    lo_sns->settopicattributes(
        iv_topicarn = iv_topic_arn
        iv_attributename = iv_attribute_name
        iv_attributevalue = iv_attribute_value
    ).
    MESSAGE 'Set/Updated SNS topic attributes' TYPE 'I'.
CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist' TYPE 'E'.
ENDTRY.
```

- For API details, see [SetTopicAttributes](#) in *AWS SDK for SAP ABAP API reference*.

## Subscribe an email address to a topic

The following code example shows how to subscribe an email address to an Amazon SNS topic.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sns->subscribe(  
        "oo_result is returned for  
        testing purpose"  
        iv_topicarn = iv_topic_arn  
        iv_protocol = 'email'  
        iv_endpoint = iv_email_address  
        iv_returnsubscriptionarn = abap_true  
    ).  
    MESSAGE 'Email address subscribed to SNS topic' TYPE 'I'.  
    CATCH /aws1/cx_snsnotfoundexception.  
        MESSAGE 'Topic does not exist' TYPE 'E'.  
    CATCH /aws1/cx_snssubscriptionlmte00.  
        MESSAGE 'Unable to create subscriptions, you have reached the maximum number of  
        subscriptions allowed.' TYPE 'E'.  
    ENDTRY.
```

- For API details, see [Subscribe](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios

### Create and publish to a FIFO topic

The following code example shows how to create and publish to a FIFO Amazon SNS topic.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a FIFO topic, subscribe an Amazon SQS FIFO queue to the topic, and publish a message to an Amazon SNS topic.

```
" Creates a FIFO topic "  
DATA lt_tpc_attributes TYPE /aws1/cl_snstopicattrsmapping_w=>tt_topicattributesmap.
```

```

        DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicatrrsmap_w=>ts_topicattributesmap_maprow.
        ls_tpc_attributes-key = 'FifoTopic'.
        ls_tpc_attributes-value = NEW /aws1/cl_snstopicatrrsmap_w( iv_value = 'true' ).
INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

TRY.
    DATA(lo_create_result) = lo_sns->createtopic(
        iv_name = iv_topic_name
        it_attributes = lt_tpc_attributes
    ).
    DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).           " ov_topic_arn
is returned for testing purpose "
    MESSAGE 'FIFO topic created' TYPE 'I'.
    CATCH /aws1/cx_snstopiclimitecdex.
        MESSAGE 'Unable to create more topics as you have reached the maximum number of
topics allowed.' TYPE 'E'.
ENDTRY.

" Subscribes an endpoint to an Amazon SNS topic "
" Only SQS FIFO queues can be subscribed to an SNS FIFO topic "
TRY.
    DATA(lo_subscribe_result) = lo_sns->subscribe(
        iv_topicarn = lv_topic_arn
        iv_protocol = 'sqS'
        iv_endpoint = iv_queue_arn
    ).
    DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).           " ov_subscription_arn
is returned for testing purpose "
    MESSAGE 'SQS Queue was subscribed to SNS topic' TYPE 'I'.
    CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist' TYPE 'E'.
    CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions, you have reached the maximum number of
subscriptions allowed.' TYPE 'E'.
ENDTRY.

" Publish message to SNS topic "
TRY.
    DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
    DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
'String' iv_stringvalue = 'High' ).
INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
            iv_topicarn = lv_topic_arn
            iv_message = 'The price of your mobile plan has been increased from $19 to
$23'
            iv_subject = 'Changes to mobile plan'
            iv_messagegroupid = 'Update-2'
            iv_messagededuplicationid = 'Update-2.1'
            it_messageattributes = lt_msg_attributes
        ).           " ov_message_id
is returned for testing purpose "
        MESSAGE 'Message was published to SNS topic' TYPE 'I'.
        CATCH /aws1/cx_snsnotfoundexception.
            MESSAGE 'Topic does not exist' TYPE 'E'.
ENDTRY.

```

## Amazon SQS examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon Simple Queue Service.

**Actions** are code excerpts that show you how to call individual service functions.

**Scenarios** are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4528\)](#)

## Actions

### Create a queue

The following code example shows how to create an Amazon SQS queue.

#### SDK for SAP ABAP

##### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an Amazon SQS standard queue.

```
TRY.  
    oo_result = lo_sqs->createqueue( iv_queuename = iv_queue_name ).           "  
oo_result is returned for testing purpose "  
    MESSAGE 'SQS queue created' TYPE 'I'.  
    CATCH /aws1/cx_sqssqueuedeldrecently.  
        MESSAGE 'After deleting a queue, you should wait 60 seconds before creating  
another queue with the same name.' TYPE 'E'.  
        CATCH /aws1/cx_sqssqueuenameexists.  
            MESSAGE 'A queue with this name already exists.' TYPE 'E'.  
        ENDTRY.
```

Create an Amazon SQS queue that waits for a message to arrive.

```
TRY.  
    DATA lt_attributes TYPE /aws1/cl_sqssqueueattrmap_w=>tt_queueattributemap.  
    DATA ls_attribute TYPE /aws1/cl_sqssqueueattrmap_w=>ts_queueattributemap_maprow.  
    ls_attribute-key = 'ReceiveMessageWaitTimeSeconds'.                      " time in  
seconds for long polling i.e. time for which the call waits for a message to arrive in  
queue before returning"  
    ls_attribute-value = NEW /aws1/cl_sqssqueueattrmap_w( iv_value = iv_wait_time ).  
    INSERT ls_attribute INTO TABLE lt_attributes.  
    oo_result = lo_sqs->createqueue(                                     " oo_result is returned for  
testing purpose "  
        iv_queuename = iv_queue_name  
        it_attributes = lt_attributes
```

```
).
MESSAGE 'SQS queue created' TYPE 'I'.
CATCH /aws1/cx_sqqueuedelrecently.
MESSAGE 'After deleting a queue, you should wait 60 seconds before creating
another queue with the same name.' TYPE 'E'.
CATCH /aws1/cx_sqsqqueuenameexists.
MESSAGE 'A queue with this name already exists.' TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateQueue](#) in *AWS SDK for SAP ABAP API reference*.

## Delete a queue

The following code example shows how to delete an Amazon SQS queue.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
lo_sqs->deletequeue( iv_queueurl = iv_queue_url ).
MESSAGE 'SQS queue deleted' TYPE 'I'.
ENDTRY.
```

- For API details, see [DeleteQueue](#) in *AWS SDK for SAP ABAP API reference*.

## Get the URL of a queue

The following code example shows how to get the URL of an Amazon SQS queue.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
oo_result = lo_sqs->getqueueurl( iv_queuename = iv_queue_name ).           "
oo_result is returned for testing purpose "
MESSAGE 'Queue URL retrieved' TYPE 'I'.
CATCH /aws1/cx_sqqueuedoesnotexist.
MESSAGE 'The requested queue does not exist' TYPE 'E'.
ENDTRY.
```

- For API details, see [GetQueueUrl](#) in *AWS SDK for SAP ABAP API reference*.

## List queues

The following code example shows how to list Amazon SQS queues.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.  
    oo_result = lo_sqs->listqueues( ).           " oo_result is returned for testing  
purpose "  
    MESSAGE 'Retrieved list of queue(s)' TYPE 'I'.  
ENDTRY.
```

- For API details, see [ListQueues](#) in *AWS SDK for SAP ABAP API reference*.

## Receive messages from a queue

The following code example shows how to receive messages from an Amazon SQS queue.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Receive a message from an Amazon SQS queue.

```
TRY.  
    oo_result = lo_sqs->receivemessage( iv_queueurl = iv_queue_url ).      "  
oo_result is returned for testing purpose "  
    DATA(lt_messages) = oo_result->get_messages( ).  
    MESSAGE 'Message received from SQS queue' TYPE 'I'.  
    CATCH /aws1/cx_sqsoverlimit.  
        MESSAGE 'Maximum number of in flight messages reached' TYPE 'E'.  
    ENDTRY.
```

Receive a message from an Amazon SQS queue using long-poll support.

```
TRY.  
    oo_result = lo_sqs->receivemessage(                  " oo_result is returned for  
testing purpose "  
        iv_queueurl = iv_queue_url  
        iv_waittimeseconds = iv_wait_time      " time in seconds for long  
polling i.e. time for which the call waits for a message to arrive in queue before  
returning "  
    ).  
    DATA(lt_messages) = oo_result->get_messages( ).
```

```
MESSAGE 'Message received from SQS queue' TYPE 'I'.
CATCH /aws1/cx_sqsoverlimit.
MESSAGE 'Maximum number of in flight messages reached' TYPE 'E'.
ENDTRY.
```

- For API details, see [ReceiveMessage in AWS SDK for SAP ABAP API reference](#).

## Send a message to a queue

The following code example shows how to send a message to an Amazon SQS queue.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_sqs->sendmessage(                                     " oo_result is returned for
testing purpose "
        iv_queueurl = iv_queue_url
        iv_messagebody = iv_message
    ).
    MESSAGE 'Message sent to SQS queue' TYPE 'I'.
CATCH /aws1/cx_sqsinvalidmsgconts.
    MESSAGE 'Message contains invalid characters' TYPE 'E'.
CATCH /aws1/cx_sqsunsupportedop.
    MESSAGE 'Operation not supported' TYPE 'E'.
ENDTRY.
```

- For API details, see [SendMessage in AWS SDK for SAP ABAP API reference](#).

## SageMaker examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon SageMaker.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4531\)](#)
- [Scenarios \(p. 4540\)](#)

## Actions

### Create a model

The following code example shows how to create a model in SageMaker.

## SDK for SAP ABAP

### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_primarycontainer TYPE REF TO /aws1/cl_sgmcontainerdefn.

"Create an ABAP object for the container image based on input variables."
CREATE OBJECT lo_primarycontainer
  EXPORTING
    iv_image      = iv_container_image
    iv_modeldataurl = iv_model_data_url.

"Create a Sagemaker model"
TRY.
  oo_result = lo_sgm->createmodel(           " oo_result is returned for testing
purpose "
    iv_executionrolearn = iv_execution_role_arn
    iv_modelname = iv_model_name
    lo_primarycontainer = lo_primarycontainer
  ).
  MESSAGE 'Model created' TYPE 'I'.
CATCH /aws1/cx_sgmresourcelimitexcd.
  MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateModel](#) in *AWS SDK for SAP ABAP API reference*.

## Create an endpoint

The following code example shows how to create a SageMaker endpoint.

## SDK for SAP ABAP

### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lt_production_variants TYPE /aws1/
cl_sgmproductionvariant=>tt_productionvariantlist.
DATA lo_production_variants TYPE REF TO /aws1/cl_sgmproductionvariant.
DATA oo_ep_config_result TYPE REF TO /aws1/cl_sgmcreateendptcfgout.

"Create a production variant as an ABAP object"
"Identifies a model that you want to host and the resources chosen to deploy for
hosting it."
CREATE OBJECT lo_production_variants
  EXPORTING
    iv_variantname      = iv_variant_name
    iv_modelname        = iv_model_name
```

```
    iv_initialinstancecount = iv_initial_instance_count
    iv_instancetype          = iv_instance_type.

    INSERT lo_production_variants INTO TABLE lt_production_variants.

    "Create an endpoint configuration"
    TRY.
        oo_ep_config_result = lo_sgm->createendpointconfig(
            iv_endpointconfigname = iv_endpoint_config_name
            it_productionvariants = lt_production_variants
        ).
        MESSAGE 'Endpoint configuration created' TYPE 'I'.
    CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
    ENDTRY.

    "Create an endpoint"
    TRY.
        oo_result = lo_sgm->createendpoint(      " oo_result is returned for testing
purpose "
            iv_endpointconfigname = iv_endpoint_config_name
            iv_endpointname = iv_endpoint_name
        ).
        MESSAGE 'Endpoint created' TYPE 'I'.
    CATCH /aws1/cx_sgmresourcelimitexcd.
        MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
    ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [CreateEndpoint](#)
  - [CreateEndpointConfig](#)

## Delete a model

The following code example shows how to delete a model in SageMaker.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    lo_sgm->deletemodel(
        iv_modelname = iv_model_name
    ).
    MESSAGE 'Model deleted' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DeleteModel](#) in *AWS SDK for SAP ABAP API reference*.

## Delete an endpoint

The following code example shows how to delete a SageMaker endpoint.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Delete an endpoint"
TRY.
    lo_sgm->deleteendpoint(
        iv_endpointname = iv_endpoint_name
    ).
    MESSAGE 'Endpoint configuration deleted' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lv_endpoint_exception).
        DATA(lv_endpoint_error) = |"{ lo_endpoint_exception->av_err_code }" -
{ lo_endpoint_exception->av_err_msg }|.
        MESSAGE lv_endpoint_error TYPE 'E'.
ENDTRY.

"Delete an endpoint configuration"
TRY.
    lo_sgm->deleteendpointconfig(
        iv_endpointconfigname = iv_endpoint_config_name
    ).
    MESSAGE 'Endpoint deleted' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lv_endpointconfig_exception).
        DATA(lv_endpointconfig_error) = |"{ lo_endpointconfig_exception->av_err_code }" -
{ lo_endpointconfig_exception->av_err_msg }|.
        MESSAGE lv_endpointconfig_error TYPE 'E'.
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [DeleteEndpoint](#)
  - [DeleteEndpointConfig](#)

## Describe a training job

The following code example shows how to describe a SageMaker training job.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_sgm->describetrainingjob(          " oo_result is returned for
testing purpose "
        iv_trainingjobname = iv_training_job_name
```

```
).
MESSAGE 'Retrieved description of training job' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
  DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [DescribeTrainingJob](#) in *AWS SDK for SAP ABAP API reference*.

## List models

The following code example shows how to list models in SageMaker.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  oo_result = lo_sgm->listmodels(          " oo_result is returned for testing
purpose "
    iv_namecontains = iv_name_contains
  ).
  MESSAGE 'Retrieved list of models' TYPE 'I'.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
      MESSAGE lv_error TYPE 'E'.
  ENDTRY.
```

- For API details, see [ListModels](#) in *AWS SDK for SAP ABAP API reference*.

## List notebook instances

The following code example shows how to list SageMaker notebook instances.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
  oo_result = lo_sgm->listnotebookinstances(          " oo_result is returned for
testing purpose "
    iv_namecontains = iv_name_contains
  ).
  MESSAGE 'Retrieved list of notebook instances' TYPE 'I'.
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
```

```
DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.
MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [ListNotebookInstances](#) in *AWS SDK for SAP ABAP API reference*.

## List the machine learning algorithms

The following code example shows how to list SageMaker machine learning algorithms.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_sgm->listalgorithms(          " oo_result is returned for testing
purpose "
        iv_namecontains = iv_name_contains
    ).
    MESSAGE 'Retrieved list of algorithms' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- For API details, see [ListAlgorithms](#) in *AWS SDK for SAP ABAP API reference*.

## List training jobs

The following code example shows how to list SageMaker training jobs.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
TRY.
    oo_result = lo_sgm->listtrainingjobs(          " oo_result is returned for testing
purpose "
        iv_namecontains = iv_name_contains
        iv_maxresults = iv_max_results
    ).
    MESSAGE 'Retrieved list of training jobs' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.
```

```
MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- For API details, see [ListTrainingJobs](#) in *AWS SDK for SAP ABAP API reference*.

## Start a training job

The following code example shows how to start a SageMaker training job.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_hyperparameters_w TYPE REF TO /aws1/cl_sgmhyperparameters_w.
DATA lt_hyperparameters TYPE /aws1/cl_sgmhyperparameters_w=>tt_hyperparameters.
DATA lt_input_data_config TYPE /aws1/cl_sgmchannel=>tt_inputdataconfig.
DATA lo_trn_channel TYPE REF TO /aws1/cl_sgmchannel.
DATA lo_trn_datasource TYPE REF TO /aws1/cl_sgmdatasource.
DATA lo_trn_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
DATA lo_val_channel TYPE REF TO /aws1/cl_sgmchannel.
DATA lo_val_datasource TYPE REF TO /aws1/cl_sgmdatasource.
DATA lo_val_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
DATA lo_algorithm_specification TYPE REF TO /aws1/cl_sgmalgorithmsspec.
DATA lo_resource_config TYPE REF TO /aws1/cl_sgmrsrcconfig.
DATA lo_output_data_config TYPE REF TO /aws1/cl_sgmoutputdataconfig.
DATA lo_stopping_condition TYPE REF TO /aws1/cl_sgmstoppingcondition.

"Create ABAP internal table for hyperparameters based on input variables."
"These hyperparameters are based on Amazon SageMaker built-in algorithm - XGBoost"
CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_max_depth.
  INSERT VALUE #( key = 'max_depth' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

  CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_eta.
    INSERT VALUE #( key = 'eta' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

  CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_eval_metric.
    INSERT VALUE #( key = 'eval_metric' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

  CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_scale_pos_weight.
    INSERT VALUE #( key = 'scale_pos_weight' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

  CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_subsample.
    INSERT VALUE #( key = 'subsample' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

  CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_objective.
    INSERT VALUE #( key = 'objective' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

  CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_num_round.
```

```

    INSERT VALUE #( key = 'num_round' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

    "Create ABAP objects for training data sources"
    CREATE OBJECT lo_trn_s3datasource
        EXPORTING
            iv_s3datatype          = iv_trn_data_s3datatype
            iv_s3datadistributiontype = iv_trn_data_s3datadistribution
            iv_s3uri                = iv_trn_data_s3uri.

    CREATE OBJECT lo_trn_datasource
        EXPORTING
            io_s3datasource = lo_trn_s3datasource.

    CREATE OBJECT lo_trn_channel
        EXPORTING
            iv_channelname      = 'train'
            io_datasource       = lo_trn_datasource
            iv_compressiontype = iv_trn_data_compressiontype
            iv_contenttype     = iv_trn_data_contenttype.

    INSERT lo_trn_channel INTO TABLE lt_input_data_config.

    "Create ABAP objects for validation data sources"
    CREATE OBJECT lo_val_s3datasource
        EXPORTING
            iv_s3datatype          = iv_val_data_s3datatype
            iv_s3datadistributiontype = iv_val_data_s3datadistribution
            iv_s3uri                = iv_val_data_s3uri.

    CREATE OBJECT lo_val_datasource
        EXPORTING
            io_s3datasource = lo_val_s3datasource.

    CREATE OBJECT lo_val_channel
        EXPORTING
            iv_channelname      = 'validation'
            io_datasource       = lo_val_datasource
            iv_compressiontype = iv_val_data_compressiontype
            iv_contenttype     = iv_val_data_contenttype.

    INSERT lo_val_channel INTO TABLE lt_input_data_config.

    "Create an ABAP object for algorithm specification."
    CREATE OBJECT lo_algorithm_specification
        EXPORTING
            iv_trainingimage      = iv_training_image
            iv_traininginputmode = iv_training_input_mode.

    "Create an ABAP object for resource configuration"
    CREATE OBJECT lo_resource_config
        EXPORTING
            iv_instancecount   = iv_instance_count
            iv_instancetype    = iv_instance_type
            iv_volumesizeingb = iv_volume_sizeingb.

    "Create an ABAP object for output data configuration"
    CREATE OBJECT lo_output_data_config
        EXPORTING
            iv_s3outputpath = iv_s3_output_path.

    "Create ABAP object for stopping condition"
    CREATE OBJECT lo_stopping_condition
        EXPORTING
            iv_maxruntimeinseconds = iv_max_runtime_in_seconds.

```

```

"Create a training job"
TRY.
    oo_result = lo_sgm->createtrainingjob(      " oo_result is returned for testing
purpose "
        iv_trainingjobname      = iv_training_job_name
        iv_rolearn                = iv_role_arn
        it_hyperparameters       = lt_hyperparameters
        it_inputdataconfig       = lt_input_data_config
        io_algorithmspecification = lo_algorithm_specification
        io_outputdataconfig      = lo_output_data_config
        io_resourceconfig        = lo_resource_config
        io_stoppingcondition     = lo_stopping_condition
    ).
    MESSAGE 'Training job created' TYPE 'I'.
CATCH /aws1/cx_sgmresourceinuse.
    MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcenotfound.
    MESSAGE 'Resource being access is not found.' TYPE 'E'.
CATCH /aws1/cx_sgmrsrcelimitexcd.
    MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
ENDTRY.

```

- For API details, see [CreateTrainingJob](#) in *AWS SDK for SAP ABAP API reference*.

## Start a transform job

The following code example shows how to start a SageMaker transform job.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

DATA lo_transforminput TYPE REF TO /aws1/cl_sgmtransforminput.
DATA lo_transformoutput TYPE REF TO /aws1/cl_sgmtransformoutput.
DATA lo_transformresources TYPE REF TO /aws1/cl_sgmtransformresources.
DATA lo_datasource  TYPE REF TO /aws1/cl_sgmtransformdatasrc.
DATA lo_s3datasource  TYPE REF TO /aws1/cl_sgmtransforms3datasrc.

"Create ABAP object for S3 data source"
CREATE OBJECT lo_s3datasource
    EXPORTING
        iv_s3uri      = iv_tf_data_s3uri
        iv_s3datatype = iv_tf_data_s3datatype.

"Create ABAP object for data source"
CREATE OBJECT lo_datasource
    EXPORTING
        io_s3datasource = lo_s3datasource.

"Create ABAP object for transform data source."
CREATE OBJECT lo_transforminput
    EXPORTING
        io_datasource      = lo_datasource
        iv_contenttype    = iv_tf_data_contenttype
        iv_compressiontype = iv_tf_data_compressiontype.

```

```
"Create an ABAP object for resource configuration"
CREATE OBJECT lo_transformresources
  EXPORTING
    iv_instancecount = iv_instance_count
    iv_instancetype = iv_instance_type.

"Create an ABAP object for output data configuration"
CREATE OBJECT lo_transformoutput
  EXPORTING
    iv_s3outputpath = iv_s3_output_path.

"Create a transform job"
TRY.
  oo_result = lo_sgm->createtransformjob(      " oo_result is returned for testing
purpose "
    iv_modelname = iv_tf_model_name
    iv_transformjobname = iv_tf_job_name
    io_transforminput = lo_transforminput
    io_transformoutput = lo_transformoutput
    io_transformresources = lo_transformresources
  ).
  MESSAGE 'Transform job created' TYPE 'I'.
CATCH /aws1/cx_sgmresourceinuse.
  MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcenotfound.
  MESSAGE 'Resource being access is not found.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcelimitexcd.
  MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
ENDTRY.
```

- For API details, see [CreateTransformJob](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios

### Get started with models and endpoints

The following code example shows how to:

- Start a training job.
- Create a SageMaker model.
- Create an endpoint configuration.
- Create an endpoint.
- Delete an endpoint.
- Delete an endpoint configuration.
- Delete a model.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

DATA lo_hyperparameters_w TYPE REF TO /aws1/cl_sgmhyperparameters_w.
DATA lo_trn_channel TYPE REF TO /aws1/cl_sgmchannel.
DATA lo_trn_datasource TYPE REF TO /aws1/cl_sgmdatasource.
DATA lo_trn_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
DATA lo_val_channel TYPE REF TO /aws1/cl_sgmchannel.
DATA lo_val_datasource TYPE REF TO /aws1/cl_sgmdatasource.
DATA lo_val_s3datasource TYPE REF TO /aws1/cl_sgms3datasource.
DATA lo_algorithm_specification TYPE REF TO /aws1/cl_sgmalgorithmsspec.
DATA lo_resource_config TYPE REF TO /aws1/cl_sgmrsrcresourceconfig.
DATA lo_output_data_config TYPE REF TO /aws1/cl_sgmoutputdataconfig.
DATA lo_stopping_condition TYPE REF TO /aws1/cl_sgmstoppingcondition.
DATA lo_primarycontainer TYPE REF TO /aws1/cl_sgmcntainerdefn.
DATA lo_production_variants TYPE REF TO /aws1/cl_sgmproductionvariant.
DATA lo_ep_config_result TYPE REF TO /aws1/cl_sgmcrtendptcfgout.
DATA lo_training_result TYPE REF TO /aws1/cl_sgmdscrtrnjobjrsp.
DATA lt_production_variants TYPE /aws1/
cl_sgmproductionvariant=>tt_productionvariantlist.
DATA lt_input_data_config TYPE /aws1/cl_sgmchannel=>tt_inputdataconfig.
DATA lt_hyperparameters TYPE /aws1/cl_sgmhyperparameters_w=>tt_hyperparameters.
DATA lv_model_data_url TYPE /aws1/sgmurl.

lv_model_data_url = iv_s3_output_path && iv_training_job_name && '/output/
model.tar.gz'.

"Create ABAP internal table for hyperparameters based on input variables."
"These hyperparameters are based on Amazon SageMaker built-in algorithm - XGBoost"
CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_max_depth.
INSERT VALUE #( key = 'max_depth' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_eta.
INSERT VALUE #( key = 'eta' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_eval_metric.
INSERT VALUE #( key = 'eval_metric' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_scale_pos_weight.
INSERT VALUE #( key = 'scale_pos_weight' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_subsample.
INSERT VALUE #( key = 'subsample' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_objective.
INSERT VALUE #( key = 'objective' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

CREATE OBJECT lo_hyperparameters_w EXPORTING iv_value = iv_hp_num_round.
INSERT VALUE #( key = 'num_round' value = lo_hyperparameters_w ) INTO TABLE
lt_hyperparameters.

"Create ABAP internal table for data based on input variables."
"Training data"
CREATE OBJECT lo_trn_s3datasource
EXPORTING
iv_s3datatype = iv_trn_data_s3datatype
iv_s3datadistributiontype = iv_trn_data_s3datadistribution
iv_s3uri = iv_trn_data_s3uri.

CREATE OBJECT lo_trn_datasource EXPORTING io_s3datasource = lo_trn_s3datasource.

CREATE OBJECT lo_trn_channel
EXPORTING

```

```

        iv_channelname      = 'train'
        io_datasource       = lo_trn_datasource
        iv_compressiontype = iv_trn_data_compressiontype
        iv_contenttype     = iv_trn_data_contenttype.
INSERT lo_trn_channel INTO TABLE lt_input_data_config.

"Validation data"
CREATE OBJECT lo_val_s3datasource
EXPORTING
    iv_s3datatype      = iv_val_data_s3datatype
    iv_s3datadistributiontype = iv_val_data_s3datadistribution
    iv_s3uri           = iv_val_data_s3uri.

CREATE OBJECT lo_val_datasource EXPORTING io_s3datasource = lo_val_s3datasource.

CREATE OBJECT lo_val_channel
EXPORTING
    iv_channelname      = 'validation'
    io_datasource       = lo_val_datasource
    iv_compressiontype = iv_val_data_compressiontype
    iv_contenttype     = iv_val_data_contenttype.
INSERT lo_val_channel INTO TABLE lt_input_data_config.

"Create an ABAP object for algorithm specification based on input variables."
CREATE OBJECT lo_algorithm_specification
EXPORTING
    iv_trainingimage    = iv_training_image
    iv_traininginputmode = iv_training_input_mode.

"Create an ABAP object for resource configuration"
CREATE OBJECT lo_resource_config
EXPORTING
    iv_instancecount   = iv_instance_count
    iv_instancetype    = iv_instance_type
    iv_volumesizeingb = iv_volume_sizeingb.

"Create an ABAP object for output data configuration"
CREATE OBJECT lo_output_data_config EXPORTING iv_s3outputpath = iv_s3_output_path.

"Create ABAP object for stopping condition"
CREATE OBJECT lo_stopping_condition EXPORTING iv_maxruntimeinseconds =
iv_max_runtime_in_seconds.

TRY.
    lo_sgm->createtrainingjob(
        iv_trainingjobname      = iv_training_job_name
        iv_rolearn              = iv_role_arn
        it_hyperparameters       = lt_hyperparameters
        it_inputdataconfig       = lt_input_data_config
        io_algorithmspecification = lo_algorithm_specification
        io_outputdataconfig      = lo_output_data_config
        io_resourceconfig        = lo_resource_config
        io_stoppingcondition     = lo_stopping_condition
    ).
    MESSAGE 'Training job created' TYPE 'I'.
CATCH /aws1/cx_sgmresourceinuse.
    MESSAGE 'Resource being accessed is in use.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcenotfound.
    MESSAGE 'Resource being access is not found.' TYPE 'E'.
CATCH /aws1/cx_sgmresourcelimitexcd.
    MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
ENDTRY.

"Wait for training job to be completed"
lo_training_result = lo_sgm->describetrainingjob( iv_trainingjobname =
iv_training_job_name ).
```

```

WHILE lo_training_result->get_trainingjobstatus( ) <> 'Completed'.
  IF sy-index = 30.
    EXIT.           "maximum 900 seconds"
  ENDIF.
  WAIT UP TO 30 SECONDS.
  lo_training_result = lo_sgm->describetrainingjob( iv_trainingjobname =
iv_training_job_name ).
ENDWHILE.

"Create ABAP object for the container image based on input variables."
CREATE OBJECT lo_primarycontainer
  EXPORTING
    iv_image      = iv_training_image
    iv_modeldataurl = lv_model_data_url.

"Create a Sagemaker model"
TRY.
  lo_sgm->createmodel(
    iv_executionrolearn = iv_role_arn
    iv_modelname = iv_model_name
    io_primarycontainer = lo_primarycontainer
  ).
  MESSAGE 'Model created' TYPE 'I'.
  CATCH /aws1/cx_sgmresourcelimitexcd.
    MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
ENDTRY.

"Create an endpoint production variant"
CREATE OBJECT lo_production_variants
  EXPORTING
    iv_variantname      = iv_ep_variant_name
    iv_modelname = iv_model_name
    iv_initialinstancecount = iv_ep_initial_instance_count
    iv_instancetype = iv_ep_instance_type.
  INSERT lo_production_variants INTO TABLE lt_production_variants.

TRY.
  "Create an endpoint configuration"
  lo_ep_config_result = lo_sgm->createendpointconfig(
    iv_endpointconfigname = iv_ep_cfg_name
    it_productionvariants = lt_production_variants
  ).
  MESSAGE 'Endpoint configuration created' TYPE 'I'.

  "Create an endpoint"
  oo_ep_output = lo_sgm->createendpoint(          " oo_ep_output is returned for
testing purpose "
    iv_endpointconfigname = iv_ep_cfg_name
    iv_endpointname = iv_ep_name
  ).
  MESSAGE 'Endpoint created' TYPE 'I'.
  CATCH /aws1/cx_sgmresourcelimitexcd.
    MESSAGE 'You have reached the limit on the number of resources' TYPE 'E'.
ENDTRY.

"Wait for endpoint creation to be completed"
DATA(lo_endpoint_result) = lo_sgm->describeendpoint( iv_endpointname =
iv_ep_name ).
  WHILE lo_endpoint_result->get_endpointstatus( ) <> 'InService'.
    IF sy-index = 30.
      EXIT.           "maximum 900 seconds"
    ENDIF.
    WAIT UP TO 30 SECONDS.
    lo_endpoint_result = lo_sgm->describeendpoint( iv_endpointname = iv_ep_name ).
ENDWHILE.

```

```
TRY.  
    "Delete an endpoint"  
    lo_sgm->deleteendpoint(  
        iv_endpointname = iv_ep_name  
    ).  
    MESSAGE 'Endpoint deleted' TYPE 'I'.  
  
    "Delete an endpoint configuration"  
    lo_sgm->deleteendpointconfig(  
        iv_endpointconfigname = iv_ep_cfg_name  
    ).  
    MESSAGE 'Endpoint configuration deleted' TYPE 'I'.  
  
    "Delete model"  
    lo_sgm->deletemodel(  
        iv_modelname = iv_model_name  
    ).  
    MESSAGE 'Model deleted' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_endpointconfig_exception).  
        DATA(lv_endpointconfig_error) = |"{ lo_endpointconfig_exception->av_err_code }"  
        - { lo_endpointconfig_exception->av_err_msg }|.  
        MESSAGE lv_endpointconfig_error TYPE 'E'.  
    ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.

- [CreateEndpoint](#)
- [CreateEndpointConfig](#)
- [CreateModel](#)
- [CreateTrainingJob](#)
- [DeleteEndpoint](#)
- [DeleteEndpointConfig](#)
- [DeleteModel](#)
- [DescribeEndpoint](#)
- [DescribeTrainingJob](#)

## Amazon Textract examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon Textract.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4544\)](#)
- [Scenarios \(p. 4550\)](#)

## Actions

### Analyze a document

The following code example shows how to analyze a document using Amazon Textract.

## SDK for SAP ABAP

### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Detects text and additional elements, such as forms or tables,"  
"in a local image file or from in-memory byte data."  
"The image must be in PNG or JPG format."  
  
DATA lo_document TYPE REF TO /aws1/cl_txdocument.  
DATA lo_s3object TYPE REF TO /aws1/cl_txs3object.  
DATA lo_featuretypes TYPE REF TO /aws1/cl_txfeaturetypes_w.  
DATA lt_featuretypes TYPE /aws1/cl_txfeaturetypes_w=>tt_featuretypes.  
  
"Create ABAP objects for feature type"  
"add TABLES to return information about the tables"  
"add FORMS to return detected form data."  
"To perform both types of analysis, add TABLES and FORMS to FeatureTypes"  
  
CREATE OBJECT lo_featuretypes EXPORTING iv_value = 'FORMS'.  
INSERT lo_featuretypes INTO TABLE lt_featuretypes.  
  
CREATE OBJECT lo_featuretypes EXPORTING iv_value = 'TABLES'.  
INSERT lo_featuretypes INTO TABLE lt_featuretypes.  
  
"Create an ABAP object for the S3 obejct."  
CREATE OBJECT lo_s3object  
    EXPORTING  
        iv_bucket = iv_s3bucket  
        iv_name   = iv_s3object.  
  
"Create an ABAP object for the document."  
CREATE OBJECT lo_document EXPORTING io_s3object = lo_s3object.  
  
"Analyze document stored in S3"  
TRY.  
    oo_result = lo_tx->analyzedocument(          "oo_result is returned for testing  
purpose"  
    EXPORTING  
        io_document      = lo_document  
        it_featuretypes = lt_featuretypes  
    ).  
    MESSAGE 'Analyze document completed' TYPE 'I'.  
    CATCH /aws1/cx_txaccessdeniedex .  
        MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.  
    CATCH /aws1/cx_txbaddocumentex .  
        MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.  
    CATCH /aws1/cx_txdocumenttoolargeex .  
        MESSAGE 'The document is too large' TYPE 'E'.  
    CATCH /aws1/cx_txhquotaexceededex .  
        MESSAGE 'Human loop quota has been exceeded' TYPE 'E'.  
    CATCH /aws1/cx_txinternalservererr .  
        MESSAGE 'Internal server error' TYPE 'E'.  
    CATCH /aws1/cx_txinvalidparameterex .  
        MESSAGE 'Request has invalid parameters' TYPE 'E'.  
    CATCH /aws1/cx_txinvalids3objectex .  
        MESSAGE 'S3 object is invalid' TYPE 'E'.  
    CATCH /aws1/cx_txprovthruputexdex .
```

```
MESSAGE 'Provisioned throughput exceeded limit' TYPE 'E'.
CATCH /aws1/cx_txthrottlingex .
MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
CATCH /aws1/cx_txunsupporteddocex .
MESSAGE 'The document is not supported' TYPE 'E'.
ENDTRY.
```

- For API details, see [AnalyzeDocument](#) in *AWS SDK for SAP ABAP API reference*.

## Detect text in a document

The following code example shows how to detect text in a document using Amazon Textract.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Detects text in the input document."
"Amazon Textract can detect lines of text and the words that make up a line of
text."
"The input document must be in one of the following image formats: JPEG, PNG, PDF,
or TIFF."

DATA lo_document TYPE REF TO /aws1/cl_txdocument.
DATA lo_s3object TYPE REF TO /aws1/cl_txs3object.

"Create an ABAP object for the S3 obejct."
CREATE OBJECT lo_s3object
EXPORTING
  iv_bucket = iv_s3bucket
  iv_name   = iv_s3object.

"Create an ABAP object for the document."
CREATE OBJECT lo_document EXPORTING io_s3object = lo_s3object.

"Analyze document stored in S3"
TRY.
  oo_result = lo_tx->detectdocumenttext( io_document = lo_document ).
"oo_result is returned for testing purpose"
  MESSAGE 'Detect document text completed' TYPE 'I'.
  CATCH /aws1/cx_txaccessdeniedex .
    MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
  CATCH /aws1/cx_txbaddocumentex .
    MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
  CATCH /aws1/cx_txdocumenttoolargeex .
    MESSAGE 'The document is too large' TYPE 'E'.
  CATCH /aws1/cx_txinternalservererr .
    MESSAGE 'Internal server error' TYPE 'E'.
  CATCH /aws1/cx_txinvalidparameterex .
    MESSAGE 'Request has invalid parameters' TYPE 'E'.
  CATCH /aws1/cx_txinvalids3objectex .
    MESSAGE 'S3 object is invalid' TYPE 'E'.
  CATCH /aws1/cx_txprovthrputexcde .
    MESSAGE 'Provisioned throughput exceeded limit' TYPE 'E'.
```

```
CATCH /aws1/cx_txthrottlingex .
MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
CATCH /aws1/cx_unsupporteddocex .
MESSAGE 'The document is not supported' TYPE 'E'.
ENDTRY.
```

- For API details, see [DetectDocumentText](#) in *AWS SDK for SAP ABAP API reference*.

## Get data about a document analysis job

The following code example shows how to get data about an Amazon Textract document analysis job.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Gets the results for an Amazon Textract"
"asynchronous operation that analyzes text in a document."
TRY.
    oo_result = lo_tex->getdocumentanalysis( iv_jobid = iv_jobid ).      "oo_result
is returned for testing purpose"
    MESSAGE 'Document analysis retrieved' TYPE 'I'.
    CATCH /aws1/cx_txaccessdeniedex .
        MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
    CATCH /aws1/cx_txinternalservererr .
        MESSAGE 'Internal server error' TYPE 'E'.
    CATCH /aws1/cx_txinvalidjobidex .
        MESSAGE 'Job ID is invalid' TYPE 'E'.
    CATCH /aws1/cx_txinvalidkmskeyex .
        MESSAGE 'KMS key is invalid' TYPE 'E'.
    CATCH /aws1/cx_txinvalidparameterex .
        MESSAGE 'Request has invalid parameters' TYPE 'E'.
    CATCH /aws1/cx_txinvalids3objectex .
        MESSAGE 'S3 object is invalid' TYPE 'E'.
    CATCH /aws1/cx_txprovthruputexcdex .
        MESSAGE 'Provisioned throughput exceeded limit' TYPE 'E'.
    CATCH /aws1/cx_txthrottlingex .
        MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
ENDTRY.
```

- For API details, see [GetDocumentAnalysis](#) in *AWS SDK for SAP ABAP API reference*.

## Start asynchronous analysis of a document

The following code example shows how to start asynchronous analysis of a document using Amazon Textract.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

**Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Starts the asynchronous analysis of an input document for relationships"
"between detected items such as key-value pairs, tables, and selection elements."

DATA lo_documentlocation TYPE REF TO /aws1/cl_txdocumentlocation.
DATA lo_s3object TYPE REF TO /aws1/cl_txs3object.
DATA lo_featuretypes TYPE REF TO /aws1/cl_txfeaturetypes_w.
DATA lt_featuretypes TYPE /aws1/cl_txfeaturetypes_w=>tt_featuretypes.

"Create ABAP objects for feature type"
"add TABLES to return information about the tables"
"add FORMS to return detected form data."
"To perform both types of analysis, add TABLES and FORMS to FeatureTypes"

CREATE OBJECT lo_featuretypes EXPORTING iv_value = 'FORMS'.
INSERT lo_featuretypes INTO TABLE lt_featuretypes.

CREATE OBJECT lo_featuretypes EXPORTING iv_value = 'TABLES'.
INSERT lo_featuretypes INTO TABLE lt_featuretypes.

"Create an ABAP object for the S3 object."
CREATE OBJECT lo_s3object
  EXPORTING
    iv_bucket = iv_s3bucket
    iv_name   = iv_s3object.

"Create an ABAP object for the document."
CREATE OBJECT lo_documentlocation EXPORTING io_s3object = lo_s3object.

"Start async document analysis."
TRY.
  oo_result = lo_tx->startdocumentanalysis(          "oo_result is returned for
testing purpose"
    EXPORTING
      io_documentlocation      = lo_documentlocation
      it_featuretypes         = lt_featuretypes
    ).
  MESSAGE 'Document analysis started' TYPE 'I'.
CATCH /aws1/cx_txaccessdeniedex .
  MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
CATCH /aws1/cx_txbaddocumentex .
  MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
CATCH /aws1/cx_txdocumenttoolargeex .
  MESSAGE 'The document is too large' TYPE 'E'.
CATCH /aws1/cx_txidempotentprmmis00 .
  MESSAGE 'Idempotent parameter mismatch exception' TYPE 'E'.
CATCH /aws1/cx_txinternalservererr .
  MESSAGE 'Internal server error' TYPE 'E'.
CATCH /aws1/cx_txinvalidkmskeyex .
  MESSAGE 'KMS key is invalid' TYPE 'E'.
CATCH /aws1/cx_txinvalidparameterex .
  MESSAGE 'Request has invalid parameters' TYPE 'E'.
CATCH /aws1/cx_txinvalids3objectex .
  MESSAGE 'S3 object is invalid' TYPE 'E'.
CATCH /aws1/cx_txlimitexceededex .
  MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
CATCH /aws1/cx_txprovthruputexcdex .
  MESSAGE 'Provisioned throughput exceeded limit' TYPE 'E'.
CATCH /aws1/cx_txthrottlingex .
  MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
```

```
CATCH /aws1/cx_txunsupporteddocex .
MESSAGE 'The document is not supported' TYPE 'E'.
ENDTRY.
```

- For API details, see [StartDocumentAnalysis](#) in *AWS SDK for SAP ABAP API reference*.

## Start asynchronous text detection

The following code example shows how to start asynchronous text detection in a document using Amazon Textract.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Starts the asynchronous detection of text in a document."
"Amazon Textract can detect lines of text and the words that make up a line of
text."
DATA lo_documentlocation TYPE REF TO /aws1/cl_txdocumentlocation.
DATA lo_s3object TYPE REF TO /aws1/cl_txs3object.

"Create an ABAP object for the S3 obejct."
CREATE OBJECT lo_s3object
  EXPORTING
    iv_bucket = iv_s3bucket
    iv_name   = iv_s3object.

"Create an ABAP object for the document."
CREATE OBJECT lo_documentlocation
  EXPORTING
    io_s3object = lo_s3object.

"Start document analysis."
TRY.
  oo_result = lo_tx->startdocumenttextdetection( io_documentlocation =
lo_documentlocation ).           "oo_result is returned for testing purpose"
  MESSAGE 'Document analysis started' TYPE 'I'.
  CATCH /aws1/cx_txaccessdeniedex .
    MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
  CATCH /aws1/cx_txbaddocumentex .
    MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
  CATCH /aws1/cx_txdocumenttoolargeex .
    MESSAGE 'The document is too large' TYPE 'E'.
  CATCH /aws1/cx_txidempotentprmmis00 .
    MESSAGE 'Idempotent parameter mismatch exception' TYPE 'E'.
  CATCH /aws1/cx_txinternalservererr .
    MESSAGE 'Internal server error' TYPE 'E'.
  CATCH /aws1/cx_txinvalidkmskeyex .
    MESSAGE 'KMS key is invalid' TYPE 'E'.
  CATCH /aws1/cx_txinvalidparameterex .
    MESSAGE 'Request has invalid parameters' TYPE 'E'.
  CATCH /aws1/cx_txinvalids3objectex .
    MESSAGE 'S3 object is invalid' TYPE 'E'.
```

```
CATCH /aws1/cx_txlimitexceededex .
  MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
CATCH /aws1/cx_txprovthrputexcdex .
  MESSAGE 'Provisioned throughput exceeded limit' TYPE 'E'.
CATCH /aws1/cx_txthrottlingex .
  MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
CATCH /aws1/cx_txunsupporteddocex .
  MESSAGE 'The document is not supported' TYPE 'E'.
ENDTRY.
```

- For API details, see [StartDocumentTextDetection](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios

### Get started with document analysis

The following code example shows how to:

- Start asynchronous analysis.
- Get document analysis.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_documentlocation TYPE REF TO /aws1/cl_txdocumentlocation.
DATA lo_s3object TYPE REF TO /aws1/cl_txs3object.
DATA lo_featuretypes TYPE REF TO /aws1/cl_txfeaturetypes_w.
DATA lt_featuretypes TYPE /aws1/cl_txfeaturetypes_w=>tt_featuretypes.

"Create ABAP objects for feature type"
"add TABLES to return information about the tables"
"add FORMS to return detected form data."
"To perform both types of analysis, add TABLES and FORMS to FeatureTypes"

CREATE OBJECT lo_featuretypes EXPORTING iv_value = 'FORMS'.
INSERT lo_featuretypes INTO TABLE lt_featuretypes.

CREATE OBJECT lo_featuretypes EXPORTING iv_value = 'TABLES'.
INSERT lo_featuretypes INTO TABLE lt_featuretypes.

"Create an ABAP object for the S3 obejct."
CREATE OBJECT lo_s3object
  EXPORTING
    iv_bucket = iv_s3bucket
    iv_name   = iv_s3object.

"Create an ABAP object for the document."
CREATE OBJECT lo_documentlocation EXPORTING io_s3object = lo_s3object.

"Start document analysis."
TRY.
  DATA(lo_start_result) = lo_tx->startdocumentanalysis(
```

```

EXPORTING
    io_documentlocation      = lo_documentlocation
    it_featuretypes          = lt_featuretypes
).
    MESSAGE 'Document analysis started' TYPE 'I'.
CATCH /aws1/cx_txaccessdeniedex .
    MESSAGE 'You do not have permission to perform this action.' TYPE 'E'.
CATCH /aws1/cx_txbaddocumentex .
    MESSAGE 'Amazon Textract is not able to read the document.' TYPE 'E'.
CATCH /aws1/cx_txdocumenttoolargeex .
    MESSAGE 'The document is too large' TYPE 'E'.
CATCH /aws1/cx_txidempotentprmmis00 .
    MESSAGE 'Idempotent parameter mismatch exception' TYPE 'E'.
CATCH /aws1/cx_txinternalservererr .
    MESSAGE 'Internal server error' TYPE 'E'.
CATCH /aws1/cx_txinvalidkmskeyex .
    MESSAGE 'KMS key is invalid' TYPE 'E'.
CATCH /aws1/cx_txinvalidparameterex .
    MESSAGE 'Request has invalid parameters' TYPE 'E'.
CATCH /aws1/cx_txinvalids3objectex .
    MESSAGE 'S3 object is invalid' TYPE 'E'.
CATCH /aws1/cx_txlimitexceededex .
    MESSAGE 'An Amazon Textract service limit was exceeded.' TYPE 'E'.
CATCH /aws1/cx_txprovthruputexcdex .
    MESSAGE 'Provisioned throughput exceeded limit' TYPE 'E'.
CATCH /aws1/cx_txthrottlingex .
    MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
CATCH /aws1/cx_txunsupporteddocex .
    MESSAGE 'The document is not supported' TYPE 'E'.
ENDTRY.

"Get job ID from the output"
DATA(lv_jobid) = lo_start_result->get_jobid( ).

"Wait for job to complete"
oo_result = lo_tex->getdocumentanalysis( EXPORTING iv_jobid = lv_jobid ).      "
oo_result is returned for testing purpose "
WHILE oo_result->get_jobstatus( ) <> 'SUCCEEDED'.
    IF sy-index = 10.
        EXIT.                      "maximum 300 seconds"
    ENDIF.
    WAIT UP TO 30 SECONDS.
    oo_result = lo_tex->getdocumentanalysis( EXPORTING iv_jobid = lv_jobid ).
ENDWHILE.

```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [GetDocumentAnalysis](#)
  - [StartDocumentAnalysis](#)

## Amazon Translate examples using SDK for SAP ABAP

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for SAP ABAP with Amazon Translate.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4552\)](#)

- [Scenarios \(p. 4555\)](#)

## Actions

### Describe a translation job

The following code example shows how to describe an Amazon Translate translation job.

#### SDK for SAP ABAP

##### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Gets the properties associated with an asynchronous batch translation job"
"including name, ID, status, source and target languages, input/output S3 buckets,
and so on."
TRY.
    oo_result = lo_xl8->describetexttranslationjob(          "oo_result is returned for
testing purpose"
        EXPORTING
            iv_jobid      = iv_jobid
        ).
    MESSAGE 'Job description retrieved' TYPE 'I'.
CATCH /aws1/cx_xl8internalserverex .
    MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
CATCH /aws1/cx_xl8resourcenotfoundex .
    MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
CATCH /aws1/cx_xl8toomanyrequestsex.
    MESSAGE 'You have made too many requests within a short period of time.' TYPE
'E'.
ENDTRY.
```

- For API details, see [DescribeTextTranslationJob](#) in *AWS SDK for SAP ABAP API reference*.

### List translation jobs

The following code example shows how to list the Amazon Translate translation jobs.

#### SDK for SAP ABAP

##### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Gets a list of the batch translation jobs that you have submitted."
DATA lo_filter TYPE REF TO /aws1/cl_xl8textxlationjobfilt.
"Create an ABAP object for filtering using jobname"
```

```
CREATE OBJECT lo_filter
EXPORTING
    iv_jobname = iv_jobname.

TRY.
    oo_result = lo_xl8->listtexttranslationjobs(          "oo_result is returned for
testing purpose"
        EXPORTING
            io_filter      = lo_filter
        ).
    MESSAGE 'Jobs retrieved' TYPE 'I'.
CATCH /aws1/cx_xl8internalserverex .
    MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
CATCH /aws1/cx_xl8invalidfilterex .
    MESSAGE 'The filter specified for the operation is not valid. Specify a
different filter.' TYPE 'E'.
CATCH /aws1/cx_xl8invalidrequestex .
    MESSAGE 'The request that you made is not valid.' TYPE 'E'.
CATCH /aws1/cx_xl8toomanyrequestsex.
    MESSAGE 'You have made too many requests within a short period of time.' TYPE
'E'.
ENDTRY.
```

- For API details, see [ListTextTranslationJobs](#) in *AWS SDK for SAP ABAP API reference*.

## Start a translation job

The following code example shows how to start an Amazon Translate translation job.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Starts an asynchronous batch translation job."
"Use batch translation jobs to translate large volumes of text across multiple
documents at once."

DATA lo_inputdataconfig  TYPE REF TO /aws1/cl_xl8inputdataconfig.
DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

"Create an ABAP object for the input data config"
CREATE OBJECT lo_inputdataconfig
EXPORTING
    iv_s3uri      = iv_input_data_s3uri
    iv_contenttype = iv_input_data_contenttype.

"Create an ABAP object for the output data config"
CREATE OBJECT lo_outputdataconfig
EXPORTING
    iv_s3uri = iv_output_data_s3uri.

"Create an interal table for target languages"
CREATE OBJECT lo_targetlanguagecodes
```

```

EXPORTING
    iv_value = iv_targetlanguagecode.
INSERT lo_targetlanguagecodes INTO TABLE lt_targetlanguagecodes.

TRY.
    oo_result = lo_xl8->starttexttranslationjob(          "oo_result is returned for
testing purpose"
        EXPORTING
            io_inputdataconfig = lo_inputdataconfig
            io_outputdataconfig = lo_outputdataconfig
            it_targetlanguagecodes = lt_targetlanguagecodes
            iv_dataaccessrolearn = iv_dataaccessrolearn
            iv_jobname = iv_jobname
            iv_sourcelanguagecode = iv_sourcelanguagecode
        ).
    MESSAGE 'Translation job started' TYPE 'I'.
CATCH /aws1/cx_xl8internalserverex .
    MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
CATCH /aws1/cx_xl8invparamvalueex .
    MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
CATCH /aws1/cx_xl8invalidrequestex .
    MESSAGE 'The request that you made is not valid.' TYPE 'E'.
CATCH /aws1/cx_xl8resourcenotfoundex .
    MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
CATCH /aws1/cx_xl8toomanyrequestsex .
    MESSAGE 'You have made too many requests within a short period of time.' TYPE
'E'.
CATCH /aws1/cx_xl8unsuppedlanguage00 .
    MESSAGE 'Amazon Translate does not support translation from the language of the
source text into the requested target language.' TYPE 'E'.
ENDTRY.

```

- For API details, see [StartTextTranslationJob](#) in *AWS SDK for SAP ABAP API reference*.

## Stop a translation job

The following code example shows how to stop an Amazon Translate translation job.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

"Stops an asynchronous batch translation job that is in progress."

TRY.
    oo_result = lo_xl8->stoptexttranslationjob(          "oo_result is returned for
testing purpose"
        EXPORTING
            iv_jobid      = iv_jobid
        ).
    MESSAGE 'Translation job stopped' TYPE 'I'.
CATCH /aws1/cx_xl8internalserverex .
    MESSAGE 'An internal server error occurred.' TYPE 'E'.
CATCH /aws1/cx_xl8resourcenotfoundex .
    MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
CATCH /aws1/cx_xl8toomanyrequestsex.

```

```
MESSAGE 'You have made too many requests within a short period of time.' TYPE
'E'.
ENDTRY.
```

- For API details, see [StopTextTranslationJob](#) in *AWS SDK for SAP ABAP API reference*.

## Translate text

The following code example shows how to translate text with Amazon Translate.

### SDK for SAP ABAP

#### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
"Translates input text from the source language to the target language."
TRY.
    oo_result = lo_xl8->translatetext(          "oo_result is returned for testing
purpose"
        EXPORTING
            iv_text      = iv_text
            iv_sourcelanguagecode = iv_sourcelanguagecode
            iv_targetlanguagecode = iv_targetlanguagecode
        ).
    MESSAGE 'Translation completed' TYPE 'I'.
    CATCH /aws1/cx_xl8detectedlanguage00 .
        MESSAGE 'The confidence that Amazon Comprehend accurately detected the source
language is low.' TYPE 'E'.
        CATCH /aws1/cx_xl8internalserverex .
            MESSAGE 'An internal server error occurred.' TYPE 'E'.
            CATCH /aws1/cx_xl8invalidrequestex .
                MESSAGE 'The request that you made is not valid.' TYPE 'E'.
                CATCH /aws1/cx_xl8resourcenotfoundex .
                    MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
                    CATCH /aws1/cx_xl8serviceunavailex .
                        MESSAGE 'The Amazon Translate service is temporarily unavailable.' TYPE 'E'.
                        CATCH /aws1/cx_xl8textsizeilmtdex .
                            MESSAGE 'The size of the text you submitted exceeds the size limit. ' TYPE 'E'.
                            CATCH /aws1/cx_xl8toomanyrequestsex .
                                MESSAGE 'You have made too many requests within a short period of time.' TYPE
'E'.
                                CATCH /aws1/cx_xl8unsuppledlanguage00 .
                                    MESSAGE 'Amazon Translate does not support translation from the language of the
source text into the requested target language. ' TYPE 'E'.
                            ENDTRY.
```

- For API details, see [TranslateText](#) in *AWS SDK for SAP ABAP API reference*.

## Scenarios

### Get started with translate jobs

The following code example shows how to:

- Start an asynchronous batch translation job.
- Wait for the asynchronous job to complete.
- Describe the asynchronous job.

## SDK for SAP ABAP

### Note

This documentation is for an SDK in developer preview release. The SDK is subject to change and is not recommended for use in production.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
DATA lo_inputdataconfig TYPE REF TO /aws1/cl_xl8inputdataconfig.
DATA lo_outputdataconfig TYPE REF TO /aws1/cl_xl8outputdataconfig.
DATA lt_targetlanguagecodes TYPE /aws1/
cl_xl8tgtlanguagecodes00=>tt_targetlanguagecodestrlist.
DATA lo_targetlanguagecodes TYPE REF TO /aws1/cl_xl8tgtlanguagecodes00.

"Create an ABAP object for the input data config"
CREATE OBJECT lo_inputdataconfig
  EXPORTING
    iv_s3uri      = iv_input_data_s3uri
    iv_contenttype = iv_input_data_contenttype.

"Create an ABAP object for the output data config"
CREATE OBJECT lo_outputdataconfig
  EXPORTING
    iv_s3uri = iv_output_data_s3uri.

"Create an interal table for target languages"
CREATE OBJECT lo_targetlanguagecodes
  EXPORTING
    iv_value = iv_targetlanguagecode.
INSERT lo_targetlanguagecodes INTO TABLE lt_targetlanguagecodes.

TRY.
  DATA(lo_translationjob_result) = lo_xl8->starttexttranslationjob(
    EXPORTING
      io_inputdataconfig = lo_inputdataconfig
      io_outputdataconfig = lo_outputdataconfig
      it_targetlanguagecodes = lt_targetlanguagecodes
      iv_dataaccessrolearn = iv_dataaccessrolearn
      iv_jobname = iv_jobname
      iv_sourcelanguagecode = iv_sourcelanguagecode
    ).
  MESSAGE 'Translation job started' TYPE 'I'.
  CATCH /aws1/cx_xl8internalserverex .
    MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
  CATCH /aws1/cx_xl8invparamvalueex .
    MESSAGE 'The value of the parameter is not valid.' TYPE 'E'.
  CATCH /aws1/cx_xl8invalidrequestex .
    MESSAGE 'The request that you made is not valid.' TYPE 'E'.
  CATCH /aws1/cx_xl8resourcenotfoundex .
    MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
  CATCH /aws1/cx_xl8toomanyrequestsex .
    MESSAGE 'You have made too many requests within a short period of time. ' TYPE 'E'.
  CATCH /aws1/cx_xl8unsuppedlanguage00 .
    MESSAGE 'Amazon Translate does not support translation from the language of the
source text into the requested target language.' TYPE 'E'.
```

```
ENDTRY.

"Get the job ID"
DATA(lv_jobid) = lo_translationjob_result->get_jobid( ).

"Wait for translate job to complete"
DATA(lo_des_translation_result) = lo_xl8->describetexttranslationjob( iv_jobid =
lv_jobid ).
WHILE lo_des_translation_result->get_textxlationjobproperties( )->get_jobstatus( )
<> 'COMPLETED'.
  IF sy-index = 30.
    EXIT.                      "maximum 900 seconds"
  ENDIF.
  WAIT UP TO 30 SECONDS.
  lo_des_translation_result = lo_xl8->describetexttranslationjob( iv_jobid =
lv_jobid ).
ENDWHILE.

TRY.
  oo_result = lo_xl8->describetexttranslationjob(          "oo_result is returned for
testing purpose"
    EXPORTING
      iv_jobid      = lv_jobid
    ).
  MESSAGE 'Job description retrieved' TYPE 'I'.
CATCH /aws1/cx_xl8internalserverex .
  MESSAGE 'An internal server error occurred. Retry your request.' TYPE 'E'.
CATCH /aws1/cx_xl8resourcenotfoundex .
  MESSAGE 'The resource you are looking for has not been found.' TYPE 'E'.
CATCH /aws1/cx_xl8toomanyrequestsex.
  MESSAGE 'You have made too many requests within a short period of time.' TYPE
'E'.
ENDTRY.
```

- For API details, see the following topics in *AWS SDK for SAP ABAP API reference*.
  - [DescribeTextTranslationJob](#)
  - [StartTextTranslationJob](#)

## Code examples for SDK for Swift

The code examples in this topic show you how to use the AWS SDK for Swift with AWS.

### Examples

- [Single-service actions and scenarios using SDK for Swift \(p. 4557\)](#)

## Single-service actions and scenarios using SDK for Swift

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for Swift with AWS services.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Services

- [Amazon Cognito Identity examples using SDK for Swift \(p. 4558\)](#)
- [IAM examples using SDK for Swift \(p. 4560\)](#)
- [Amazon S3 examples using SDK for Swift \(p. 4568\)](#)

# Amazon Cognito Identity examples using SDK for Swift

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Swift with Amazon Cognito Identity.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

## Topics

- [Actions \(p. 4558\)](#)

## Actions

### Create an identity pool

The following code example shows how to find the Amazon Cognito Identity with the given name, creating it if it's not found.

#### SDK for Swift

##### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a new identity pool.

```
func createIdentityPool(name: String) async throws -> String? {
    let cognitoInputCall = CreateIdentityPoolInput(developerProviderName:
"com.exampleco.CognitoIdentityDemo",
                                                identityPoolName: name)

    do {
        let result = try await cognitoIdentityClient.createIdentityPool(input:
cognitoInputCall)
        guard let poolId = result.identityPoolId else {
            return nil
        }

        return poolId
    } catch {
        print("ERROR: ", dump(error, name: "Error attempting to create the identity
pool"))
    }

    return nil
}
```

- For more information, see [AWS SDK for Swift developer guide](#).
- For API details, see the following topics in *AWS SDK for Swift API reference*.
  - [createIdentityPool](#)
  - [listIdentityPools](#)

## List identity pools

The following code example shows how to get a list of Amazon Cognito Identity pools.

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Find the ID of an identity pool given its name.

```
func getIdentityPoolID(name: String) async throws -> String? {
    var token: String? = nil

    // Iterate over the identity pools until a match is found.
    repeat {
        /// `token` is a value returned by `ListIdentityPools()` if the returned
        list
        /// of identity pools is only a partial list. You use the `token` to tell
        Cognito that
        /// you want to continue where you left off previously; specifying `nil` or
        not providing
        /// it means "start at the beginning."

        let listPoolsInput = ListIdentityPoolsInput(maxResults: 25, nextToken:
        token)

        /// Read pages of identity pools from Cognito until one is found
        /// whose name matches the one specified in the `name` parameter.
        /// Return the matching pool's ID. Each time we ask for the next
        /// page of identity pools, we pass in the token given by the
        /// previous page.

        do {
            let output = try await cognitoIdentityClient.listIdentityPools(input:
            listPoolsInput)

            if let identityPools = output.identityPools {
                for pool in identityPools {
                    if pool.identityPoolName == name {
                        return pool.identityPoolId!
                    }
                }
            }

            token = output.nextToken
        } catch {
            print("ERROR: ", dump(error, name: "Trying to get list of identity
            pools"))
        }
    } while token != nil

    return nil
}
```

Get the ID of an existing identity pool or create it if it doesn't already exist.

```
public func getOrCreateIdentityPoolID(name: String) async throws -> String? {
    // See if the pool already exists

    do {
        guard let poolId = try await self.getIdentityPoolID(name: name) else {
            let poolId = try await self.createIdentityPool(name: name)
            return poolId
        }

        return poolId
    } catch {
        print("ERROR: ", dump(error, name: "Trying to get the identity pool ID"))
        return nil
    }
}
```

- For more information, see [AWS SDK for Swift developer guide](#).
- For API details, see the following topics in [AWS SDK for Swift API reference](#).
  - [createIdentityPool](#)
  - [listIdentityPools](#)

## IAM examples using SDK for Swift

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Swift with AWS Identity and Access Management.

**Actions** are code excerpts that show you how to call individual service functions.

**Scenarios** are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4560\)](#)

## Actions

### Attach a policy to a role

The following code example shows how to attach an IAM policy to a role.

#### SDK for Swift

##### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func attachRolePolicy(role: String, policyArn: String) async throws {
    let input = AttachRolePolicyInput(
```

```
        policyArn: policyArn,
        roleName: role
    )
do {
    _ = try await client.attachRolePolicy(input: input)
} catch {
    throw error
}
}
```

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Swift API reference*.

## Create a role

The following code example shows how to create an IAM role.

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func createRole(name: String, policyDocument: String) async throws -> String
{
    let input = CreateRoleInput(
        assumeRolePolicyDocument: policyDocument,
        roleName: name
    )
    do {
        let output = try await client.createRole(input: input)
        guard let role = output.role else {
            throw ServiceHandlerError.noSuchRole
        }
        guard let id = role.roleId else {
            throw ServiceHandlerError.noSuchRole
        }
        return id
    } catch {
        throw error
    }
}
```

- For API details, see [CreateRole](#) in *AWS SDK for Swift API reference*.

## Create a service-linked role

The following code example shows how to create an IAM service-linked role.

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func createServiceLinkedRole(service: String, suffix: String? = nil,  
description: String?)  
    async throws -> IamClientTypes.Role {  
    let input = CreateServiceLinkedRoleInput(  
        aWSServiceName: service,  
        customSuffix: suffix,  
        description: description  
    )  
    do {  
        let output = try await client.createServiceLinkedRole(input: input)  
        guard let role = output.role else {  
            throw ServiceHandlerError.noSuchRole  
        }  
        return role  
    } catch {  
        throw error  
    }  
}
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for Swift API reference*.

## Create a user

The following code example shows how to create an IAM user.

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func createUser(name: String) async throws -> String {  
    let input = CreateUserInput(  
        userName: name  
    )  
    do {  
        let output = try await client.createUser(input: input)  
        guard let user = output.user else {  
            throw ServiceHandlerError.noSuchUser  
        }  
        guard let id = user.userId else {  
            throw ServiceHandlerError.noSuchUser  
        }  
        return id  
    } catch {  
        throw error  
    }  
}
```

- For API details, see [CreateUser](#) in *AWS SDK for Swift API reference*.

## Get a policy

The following code example shows how to get an IAM policy.

## SDK for Swift

### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func getPolicy(arn: String) async throws -> IamClientTypes.Policy {
    let input = GetPolicyInput(
        policyArn: arn
    )
    do {
        let output = try await client.getPolicy(input: input)
        guard let policy = output.policy else {
            throw ServiceHandlerError.noSuchPolicy
        }
        return policy
    } catch {
        throw error
    }
}
```

- For API details, see [GetPolicy](#) in *AWS SDK for Swift API reference*.

## Get a role

The following code example shows how to get an IAM role.

## SDK for Swift

### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func getRole(name: String) async throws -> IamClientTypes.Role {
    let input = GetRoleInput(
        roleName: name
    )
    do {
        let output = try await client.getRole(input: input)
        guard let role = output.role else {
            throw ServiceHandlerError.noSuchRole
        }
        return role
    } catch {
        throw error
    }
}
```

- For API details, see [GetRole](#) in *AWS SDK for Swift API reference*.

## List groups

The following code example shows how to list IAM groups.

## SDK for Swift

### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func listGroups() async throws -> [String] {
    var groupList: [String] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListGroupsInput(marker: marker)
        let output = try await client.listGroups(input: input)

        guard let groups = output.groups else {
            return groupList
        }

        for group in groups {
            if let name = group.groupName {
                groupList.append(name)
            }
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return groupList
}
```

- For API details, see [ListGroups](#) in *AWS SDK for Swift API reference*.

## List inline policies for a role

The following code example shows how to list inline policies for an IAM role.

## SDK for Swift

### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func listRolePolicies(role: String) async throws -> [String] {
    var policyList: [String] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListRolePoliciesInput(
            marker: marker,
            roleName: role
        )
        let output = try await client.listRolePolicies(input: input)

        guard let policies = output.policyNames else {
```

```
        return policyList
    }

    for policy in policies {
        policyList.append(policy)
    }
    marker = output.marker
    isTruncated = output.isTruncated
} while isTruncated == true
return policyList
}
```

- For API details, see [ListRolePolicies](#) in *AWS SDK for Swift API reference*.

## List policies

The following code example shows how to list IAM policies.

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func listPolicies() async throws -> [MyPolicyRecord] {
    var policyList: [MyPolicyRecord] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListPoliciesInput(marker: marker)
        let output = try await client.listPolicies(input: input)

        guard let policies = output.policies else {
            return policyList
        }

        for policy in policies {
            guard let name = policy.policyName,
                  let id = policy.policyId,
                  let arn = policy.arn else {
                throw ServiceHandlerError.noSuchPolicy
            }
            policyList.append(MyPolicyRecord(name: name, id: id, arn: arn))
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return policyList
}
```

- For API details, see [ListPolicies](#) in *AWS SDK for Swift API reference*.

## List policies attached to a role

The following code example shows how to list policies attached to an IAM role.

## SDK for Swift

### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
/// Returns a list of AWS Identity and Access Management (IAM) policies
/// that are attached to the role.
///
/// - Parameter role: The IAM role to return the policy list for.
///
/// - Returns: An array of `IamClientTypes.AttachedPolicy` objects
/// describing each managed policy that's attached to the role.
public func listAttachedRolePolicies(role: String) async throws ->
[IamClientTypes.AttachedPolicy] {
    var policyList: [IamClientTypes.AttachedPolicy] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListAttachedRolePoliciesInput(
            marker: marker,
            roleName: role
        )
        let output = try await client.listAttachedRolePolicies(input: input)

        guard let attachedPolicies = output.attachedPolicies else {
            return policyList
        }

        for attachedPolicy in attachedPolicies {
            policyList.append(attachedPolicy)
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return policyList
}
```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for Swift API reference*.

## List roles

The following code example shows how to list IAM roles.

## SDK for Swift

### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func listRoles() async throws -> [String] {
    var roleList: [String] = []
```

```
var marker: String? = nil
var isTruncated: Bool

repeat {
    let input = ListRolesInput(marker: marker)
    let output = try await client.listRoles(input: input)

    guard let roles = output.roles else {
        return roleList
    }

    for role in roles {
        if let name = role.roleName {
            roleList.append(name)
        }
    }
    marker = output.marker
    isTruncated = output.isTruncated
} while isTruncated == true
return roleList
}
```

- For API details, see [ListRoles](#) in [AWS SDK for Swift API reference](#).

## List users

The following code example shows how to list IAM users.

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func listUsers() async throws -> [MyUserRecord] {
    var userList: [MyUserRecord] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListUsersInput(marker: marker)
        let output = try await client.listUsers(input: input)

        guard let users = output.users else {
            return userList
        }

        for user in users {
            if let id = user.userId, let name = user.userName {
                userList.append(MyUserRecord(id: id, name: name))
            }
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return userList
}
```

- For API details, see [ListUsers](#) in *AWS SDK for Swift API reference*.

## Amazon S3 examples using SDK for Swift

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Swift with Amazon Simple Storage Service.

*Actions* are code excerpts that show you how to call individual service functions.

*Scenarios* are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

### Topics

- [Actions \(p. 4568\)](#)
- [Scenarios \(p. 4572\)](#)

## Actions

### [Copy an object from one bucket to another](#)

The following code example shows how to copy an S3 object from one bucket to another.

#### SDK for Swift

##### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func copyFile(from sourceBucket: String, name: String, to destBucket: String) async throws {
    let srcUrl = ("\"sourceBucket)/
    \name").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

    let input = CopyObjectInput(
        bucket: destBucket,
        copySource: srcUrl,
        key: name
    )
    _ = try await client.copyObject(input: input)
}
```

- For API details, see [CopyObject](#) in *AWS SDK for Swift API reference*.

## [Create a bucket](#)

The following code example shows how to create an S3 bucket.

#### SDK for Swift

##### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

##### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func createBucket(name: String) async throws {
    let config = S3ClientTypes.CreateBucketConfiguration(
        locationConstraint: .usEast2
    )
    let input = CreateBucketInput(
        bucket: name,
        createBucketConfiguration: config
    )
    _ = try await client.createBucket(input: input)
}
```

- For API details, see [CreateBucket](#) in *AWS SDK for Swift API reference*.

## Delete an empty bucket

The following code example shows how to delete an empty S3 bucket.

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    _ = try await client.deleteBucket(input: input)
}
```

- For API details, see [DeleteBucket](#) in *AWS SDK for Swift API reference*.

## Delete an object

The following code example shows how to delete an S3 object.

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.deleteObject(input: input)
    }
}
```

```
        } catch {
            throw error
        }
    }
```

- For API details, see [DeleteObject](#) in *AWS SDK for Swift API reference*.

## Delete multiple objects

The following code example shows how to delete multiple objects from an S3 bucket.

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func deleteObjects(bucket: String, keys: [String]) async throws {
    let input = DeleteObjectsInput(
        bucket: bucket,
        delete: S3ClientTypes.Delete(
            objects: keys.map({ S3ClientTypes.ObjectIdentifier(key: $0) }),
            quiet: true
        )
    )

    do {
        _ = try await client.deleteObjects(input: input)
    } catch {
        throw error
    }
}
```

- For API details, see [DeleteObjects](#) in *AWS SDK for Swift API reference*.

## Get an object from a bucket

The following code example shows how to read data from an object in an S3 bucket.

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Download an object from a bucket to a local file.

```
public func downloadFile(bucket: String, key: String, to: String) async throws {
    let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

    let input = GetObjectInput(
        bucket: bucket,
```

```
        key: key
    )
    let output = try await client.getObject(input: input)

    // Get the data stream object. Return immediately if there isn't one.
    guard let body = output.body else {
        return
    }
    let data = body.toBytes().toData()
    try data.write(to: fileUrl)
}
```

Read an object into a Swift Data object.

```
public func readFile(bucket: String, key: String) async throws -> Data {
    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    let output = try await client.getObject(input: input)

    // Get the stream and return its contents in a `Data` object. If
    // there is no stream, return an empty `Data` object instead.
    guard let body = output.body else {
        return "".data(using: .utf8)!
    }
    let data = body.toBytes().toData()
    return data
}
```

- For API details, see [GetObject](#) in *AWS SDK for Swift API reference*.

## List objects in a bucket

The following code example shows how to list objects in an S3 bucket.

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
public func listBucketFiles(bucket: String) async throws -> [String] {
    let input = ListObjectsV2Input(
        bucket: bucket
    )
    let output = try await client.listObjectsV2(input: input)
    var names: [String] = []

    guard let objList = output.contents else {
        return []
    }

    for obj in objList {
        if let objName = obj.key {
            names.append(objName)
        }
    }
}
```

```
        }
    }

    return names
}
```

- For API details, see [ListObjects](#) in *AWS SDK for Swift API reference*.

## Upload an object to a bucket

The following code example shows how to upload an object to an S3 bucket.

### SDK for Swift

#### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

#### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Upload a file from local storage to a bucket.

```
public func uploadFile(bucket: String, key: String, file: String) async throws {
    let urlString = URL(fileURLWithPath: file)
    let fileData = try Data(contentsOf: urlString)
    let dataStream = ByteStream.from(data: fileData)

    let input = PutObjectInput(
        body: dataStream,
        bucket: bucket,
        key: key
    )
    _ = try await client.putObject(input: input)
}
```

Upload the contents of a Swift Data object to a bucket.

```
public func createFile(bucket: String, key: String, withData data: Data) async
throws {
    let dataStream = ByteStream.from(data: data)

    let input = PutObjectInput(
        body: dataStream,
        bucket: bucket,
        key: key
    )
    _ = try await client.putObject(input: input)
}
```

- For API details, see [PutObject](#) in *AWS SDK for Swift API reference*.

## Scenarios

### Get started with buckets and objects

The following code example shows how to:

- Create a bucket.
- Upload a file to the bucket.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the objects in a bucket.
- Delete a bucket.

## SDK for Swift

### Note

This is prerelease documentation for an SDK in preview release. It is subject to change.

### Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

A Swift class that handles calls to the SDK for Swift.

```
import Foundation
import AWSS3
import ClientRuntime
import AWSClientRuntime

/// A class containing all the code that interacts with the AWS SDK for Swift.
public class ServiceHandler {
    let client: S3Client

    /// Initialize and return a new ``ServiceHandler`` object, which is used to drive
    /// the AWS calls
    /// used for the example.
    ///
    /// - Returns: A new ``ServiceHandler`` object, ready to be called to
    /// execute AWS operations.
    public init() async {
        do {
            client = try await S3Client()
        } catch {
            print("ERROR: ", dump(error, name: "Initializing s3 client"))
            exit(1)
        }
    }

    /// Create a new user given the specified name.
    ///
    /// - Parameters:
    ///   - name: Name of the bucket to create.
    /// - Throws an exception if an error occurs.
    public func createBucket(name: String) async throws {
        let config = S3ClientTypes.CreateBucketConfiguration(
            locationConstraint: .usEast2
        )
        let input = CreateBucketInput(
            bucket: name,
            createBucketConfiguration: config
        )
        _ = try await client.createBucket(input: input)
    }

    /// Delete a bucket.
    /// - Parameter name: Name of the bucket to delete.
}
```

```
public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    _ = try await client.deleteBucket(input: input)
}

/// Upload a file from local storage to the bucket.
/// - Parameters:
///   - bucket: Name of the bucket to upload the file to.
///   - key: Name of the file to create.
///   - file: Path name of the file to upload.
public func uploadFile(bucket: String, key: String, file: String) async throws {
    let urlString = URL(fileURLWithPath: file)
    let fileData = try Data(contentsOf: urlString)
    let dataStream = ByteStream.from(data: fileData)

    let input = PutObjectInput(
        body: dataStream,
        bucket: bucket,
        key: key
    )
    _ = try await client.putObject(input: input)
}

/// Create a file in the specified bucket with the given name. The new
/// file's contents are uploaded from a `Data` object.
///
/// - Parameters:
///   - bucket: Name of the bucket to create a file in.
///   - key: Name of the file to create.
///   - data: A `Data` object to write into the new file.
public func createFile(bucket: String, key: String, withData data: Data) async
throws {
    let dataStream = ByteStream.from(data: data)

    let input = PutObjectInput(
        body: dataStream,
        bucket: bucket,
        key: key
    )
    _ = try await client.putObject(input: input)
}

/// Download the named file to the given directory on the local device.
///
/// - Parameters:
///   - bucket: Name of the bucket that contains the file to be copied.
///   - key: The name of the file to copy from the bucket.
///   - to: The path of the directory on the local device where you want to
///     download the file.
public func downloadFile(bucket: String, key: String, to: String) async throws {
    let urlString = URL(fileURLWithPath: to).appendingPathComponent(key)

    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    let output = try await client.getObject(input: input)

    // Get the data stream object. Return immediately if there isn't one.
    guard let body = output.body else {
        return
    }
    let data = body.toBytes().toData()
    try data.write(to: urlString)
```

```
}

     /// Read the specified file from the given S3 bucket into a Swift
     /// `Data` object.
     ///
     /// - Parameters:
     ///   - bucket: Name of the bucket containing the file to read.
     ///   - key: Name of the file within the bucket to read.
     ///
     /// - Returns: A `Data` object containing the complete file data.
public func readFile(bucket: String, key: String) async throws -> Data {
    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    let output = try await client.getObject(input: input)

    // Get the stream and return its contents in a `Data` object. If
    // there is no stream, return an empty `Data` object instead.
    guard let body = output.body else {
        return "".data(using: .utf8)!
    }
    let data = body.toBytes().toData()
    return data
}

     /// Copy a file from one bucket to another.
     ///
     /// - Parameters:
     ///   - sourceBucket: Name of the bucket containing the source file.
     ///   - name: Name of the source file.
     ///   - destBucket: Name of the bucket to copy the file into.
     public func copyFile(from sourceBucket: String, name: String, to destBucket:
String) async throws {
    let srcUrl = ("\"(\(sourceBucket)/"
\((name))").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

    let input = CopyObjectInput(
        bucket: destBucket,
        copySource: srcUrl,
        key: name
    )
    _ = try await client.copyObject(input: input)
}

     /// Deletes the specified file from Amazon S3.
     ///
     /// - Parameters:
     ///   - bucket: Name of the bucket containing the file to delete.
     ///   - key: Name of the file to delete.
     ///
public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.deleteObject(input: input)
    } catch {
        throw error
    }
}

     /// Returns an array of strings, each naming one file in the
     /// specified bucket.
```

```
///  
/// - Parameter bucket: Name of the bucket to get a file listing for.  
/// - Returns: An array of `String` objects, each giving the name of  
///             one file contained in the bucket.  
public func listBucketFiles(bucket: String) async throws -> [String] {  
    let input = ListObjectsV2Input(  
        bucket: bucket  
    )  
    let output = try await client.listObjectsV2(input: input)  
    var names: [String] = []  
  
    guard let objList = output.contents else {  
        return []  
    }  
  
    for obj in objList {  
        if let objName = obj.key {  
            names.append(objName)  
        }  
    }  
  
    return names  
}
```

A Swift command-line program to manage the SDK calls.

```
import Foundation  
import ServiceHandler  
import ArgumentParser  
  
/// The command-line arguments and options available for this  
/// example command.  
struct ExampleCommand: ParsableCommand {  
    @Argument(help: "Name of the S3 bucket to create")  
    var bucketName: String  
  
    @Argument(help: "Pathname of the file to upload to the S3 bucket")  
    var uploadSource: String  
  
    @Argument(help: "The name (key) to give the file in the S3 bucket")  
    var objName: String  
  
    @Argument(help: "S3 bucket to copy the object to")  
    var destBucket: String  
  
    @Argument(help: "Directory where you want to download the file from the S3 bucket")  
    var downloadDir: String  
  
    static var configuration = CommandConfiguration(  
        commandName: "s3-basics",  
        abstract: "Demonstrates a series of basic AWS S3 functions.",  
        discussion: """  
        Performs the following Amazon S3 commands:  
  
        * `CreateBucket`  
        * `PutObject`  
        * `GetObject`  
        * `CopyObject`  
        * `ListObjects`  
        * `DeleteObjects`  
        * `DeleteBucket`  
        """  
    )
```

```
    /// Called by ``main()`` to do the actual running of the AWS
    /// example.
    func runAsync() async throws {
        let serviceHandler = await ServiceHandler()

        // 1. Create the bucket.
        print("Creating the bucket \(bucketName)...")
        try await serviceHandler.createBucket(name: bucketName)

        // 2. Upload a file to the bucket.
        print("Uploading the file \(uploadSource)...")
        try await serviceHandler.uploadFile(bucket: bucketName, key: objName, file:
uploadSource)

        // 3. Download the file.
        print("Downloading the file \(objName) to \(downloadDir)...")
        try await serviceHandler.downloadFile(bucket: bucketName, key: objName, to:
downloadDir)

        // 4. Copy the file to another bucket.
        print("Copying the file to the bucket \(destBucket)...")
        try await serviceHandler.copyFile(from: bucketName, name: objName, to:
destBucket)

        // 5. List the contents of the bucket.

        print("Getting a list of the files in the bucket \(bucketName)")
        let fileList = try await serviceHandler.listBucketFiles(bucket: bucketName)
        let numFiles = fileList.count
        if numFiles != 0 {
            print("\(numFiles) file\( (numFiles > 1) ? "s" : "") in bucket
\(bucketName):")
            for name in fileList {
                print(" \(name)")
            }
        } else {
            print("No files found in bucket \(bucketName)")
        }

        // 6. Delete the objects from the bucket.

        print("Deleting the file \(objName) from the bucket \(bucketName)...")
        try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
        print("Deleting the file \(objName) from the bucket \(destBucket)...")
        try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

        // 7. Delete the bucket.
        print("Deleting the bucket \(bucketName)...")
        try await serviceHandler.deleteBucket(name: bucketName)

        print("Done.")
    }
}

// Main program entry point.
//
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        }
    }
}
```

```
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
```

- For API details, see the following topics in *AWS SDK for Swift API reference*.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)
  - [ListObjects](#)
  - [PutObject](#)